10-23-2009

Additive Latent Variable (ALV) Modeling: Assessing Variation in Intervention Impact in Randomized Field Trials

Peter Ayo Toyinbo
University of South Florida

Additive Latent Variable (ALV) Modeling:

Assessing Variation in Intervention Impact in Randomized Field Trials

by

Peter Ayo Toyinbo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Epidemiology and Biostatistics
College of Public Health
University of South Florida

Co-Major Professor: Charles Hendricks Brown, Ph.D.
Co-Major Professor: Getachew Dagne, Ph.D.
Yiliang Zhu, Ph.D.
Paul Greenbaum, Ph.D.
Wei Wang, Ph.D.

Date of Approval:
October 23, 2009

Keywords: Generalized Additive Model, EM algorithm, Monte Carlo EM, Markov Chain Monte Carlo, Metropolis-Hastings algorithm, nonlinear structural equation model

DEDICATION

To my family.

Grace, Funbi, Femi and Tomi.

ACKNOWLEDGMENTS

This work has been shaped by one of the research goals of the Prevention Science and Methodology Group (PSMG) directed by C. Hendricks Brown, Ph.D. It is supported by a training grant (Preventive Research Training in Mental Health / 5T32MH018834-19 funded by NIMH, PI Nicholas Ialongo (JHSPH); and also by a Diversity Supplement 3R01DA019984-02S1 from NIDA, Parent grant 1R01DA019984-01, PI Sheppard Kellam (American Institutes for Research). The author is indebted to his dissertation committee members for their valuable supervision, to his fellow colleagues at the U.S.F. College of Public Health for the helpful discussions at the Biostatistical Forums, and to Zilli Sloboda, Sc.D. for allowing the data for illustration to be provided through The University of Akron's Institute for Health and Social Policy (IHSP).

.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ADDITIVE LATENT VARIABLE (ALV) MODELING:

ASSESSING VARIATION IN INTERVENTION IMPACT

IN RANDOMIZED FIELD TRIALS

Peter Ayo Toyinbo

ABSTRACT

In order to personalize or tailor treatments to maximize impact among different subgroups, there is need to model not only the main effects of intervention but also the variation in intervention impact by baseline individual level risk characteristics. To this end a suitable statistical model will allow researchers to answer a major research question: who benefits or is harmed by this intervention program? Commonly in social and psychological research, the baseline risk may be unobservable and have to be estimated from observed indicators that are measured with errors; also it may have nonlinear relationship with the outcome. Most of the existing nonlinear structural equation models (SEM's) developed to address such problems employ polynomial or fully parametric nonlinear functions to define the structural equations. These methods are limited because they require functional forms to be specified beforehand and even if the models include higher order polynomials there may be problems when the focus of interest relates to the function over its whole domain.

To develop a more flexible statistical modeling technique for assessing complex relationships between a proximal/distal outcome and 1) baseline characteristics measured with errors, and 2) baseline-treatment interaction; such that the shapes of these relationships are data driven and there is no need for the shapes to be determined a priori. **I**n the ALV model structure the nonlinear components of the regression equations are represented as generalized additive model (GAM), or generalized additive mixed-effects model (GAMM).

Replication study results show that the ALV model estimates of underlying relationships in the data are sufficiently close to the true pattern. The ALV modeling technique allows researchers to assess how an intervention affects individuals differently as a function of baseline risk that is itself measured with error, and uncover complex relationships in the data that might otherwise be missed. Although the ALV approach is computationally intensive, it relieves its users from the need to decide functional forms before the model is run. It can be extended to examine complex nonlinearity between growth factors and distal outcomes in a longitudinal study.

CHAPTER 1

INTRODUCTION

1.1 Variation in Intervention Impact across Individual Level Baseline Characteristics

Preventive interventions focus on reducing specific risk factors or enhancing protective factors. Intervention theory posits that a distal target or an outcome should be impacted by effectively modifying the risk factors. The success of a typical intervention program then is measured in terms of how well it impacts distal preventive targets. In an experimental design aimed at assessing the effects of an intervention, randomization is carried out to ensure that all the intervention groups are comparable at baseline with respect to the risk factors and other systematic effects. Given a successful randomization, post intervention differences between the groups may be attributable solely to intervention effects. However even in randomized trials there remains a variable degree of within-group heterogeneity with respect to individual level baseline risk that can potentially modify the intervention impact. Generally such variability is more common in field trials that are universal than in standard efficacy trials. Depending on the nature of an intervention program, its effects on a risk factor may be expected to vary according to the individual's baseline status on the risk scale; that is, the intervention impact may not be same for all individuals. One example is a randomized field trial (RFT) where the intervention, Good Behavior Game (GBG), was designed to reduce aggressive behavior in first grade kids (Muthen, 2002; Kellam, et al., 2008; Brown, et al., 2008; Poduska, et al., 2008). The investigators predicted that, since about half the kids have minimal level of aggressive behavior throughout life, GBG would have impact on only those kids who scored high on the risk scale for aggressive behavior

at baseline. These predictions can be tested statistically by measuring the significance of interaction effects between the baseline aggressive behavior and the intervention (Kellam, et al., 2008; Poduska, et al., 2008; Brown, et al., 2008).

In their paper Brown, et al. (2008) the authors argued that the focus of intent-to-treat (ITT) analyses should not be limited to that of examining the main effects of intervention. They explained that one reason to also model the variation in intervention impact by baseline individual level risk characteristics is "to personalize or tailor treatments to maximize impact among different subgroups". It is also possible for an intervention to be beneficial to some individuals while the same is harmful to others. The authors concluded that "there is almost always an *a priori* reason to examine interactions involving intervention and baseline level of risk", particularly in RFT's employing universal preventive interventions.

Often times investigators are interested in how multiple risk factors act together to predict a long term outcome. In social sciences in particular, these observable risk factors often have measurement errors. To obtain a summary risk variable for use in a statistical analysis designed to investigate intervention effect, a better alternative to simple averaging might be hypothetical latent risk variable(s) that is/are constructed from the observed baseline risk variables. The risk variables then serve as indicators for the underlying latent construct(s) or factor(s). By using e.g. a single latent summary variable, the accuracy of the results is less affected by measurement error in any individual risk factor. Multiple measures (indicators) of a construct therefore tend to be more valid and reliable than when only a single fallible indicator is used (Kline, 2005). In a more complex scenario the observed risk indicators may be jointly multi-dimensional rather than uni-dimensional; the dimension is reflected by the number of constructs underlying the observed indicators. So, when risk summary by simple averaging is not an option, a latent variable

modeling technique (such as structural equation modeling (SEM) and its extensions) is most appropriate for carrying out such analyses.

In a SEM, the observed baseline risk indicators are summarized into latent factor(s) in the measurement part of the model. Here the estimate(s) of the latent factor(s) constitutes the latent baseline risk, the common contents of the observed risk indicators. The latent baseline risk factor(s) may then be applied in the structural part of the SEM to relate to one another and also as predictor(s) of one or more distal outcomes. This approach can be extended for longitudinal data modeling where estimated latent growth factors (intercept and slope) may relate to each other and predict a later outcome in a system of linear equations within the SEM framework (Muthen, 2002). For simplicity and to keep within the scope of this dissertation, we will assume that the indicators are uni-dimensional, and the single latent factor predicts a single distal outcome in a sub-model. To assess variation in intervention impact as a function of the baseline, a dummy-coded intervention variable plus its interaction term with baseline are added to the sub-model.

The conventional SEM seeks simultaneous solutions to a system of linear regression equations that form the measurement and the structural parts, with the assumption that the latent factors (or latent baseline risk) influence the other set of variables (e.g. a distal outcome) or their transformed versions in a linear fashion. However if in reality the latent baseline relates to the outcome it predicts in a non-linear fashion, the usual linear modeling methods can easily lead to erroneous conclusion (Wang, Brown, & Bandeen-Roche, 2005). Also as a result of such misspecification, a test of intervention impact on later outcomes may result in non-significant findings due to low statistical power. Even with significant effects, there may be misinterpretation of what the effects mean. The conditions under which such intervention impacts occur may only be fully captured by methods that allow for nonlinear relations among both the

observed and unobserved variables. Linear models, even if they include higher order polynomials, do not provide sufficient flexibility in assessing this potential type of impact (Brown, 1993). So, even the advanced SEM technique that includes polynomial model specifications has its limitations.

1.2 The Concept of Additive Latent Variable (ALV) Modeling

An alternative approach to modeling nonlinear relationship is to use a nonparametric estimation method. These methods can vary in complexity. For example, an estimation method was developed for panel data (binary or continuous) with correlated errors (Chib & Jeliazkov, 2006). In the authors' regression model, the response variable depends parametrically on some covariate vectors and nonparametrically on another covariate where the relationship is unspecified but the function is assumed to be smooth, otherwise unrestricted. The authors used MCMC based algorithm in the estimation of the nonparametric function when the errors are correlated. They were able to incorporate nonlinearity and intertemporal dependence in their model (Chib & Jeliazkov, 2006). While their setting is a single regression model aimed at distinguishing among "important sources of intertemporal dependence in the observations", our focus here is on developing a system of simultaneous regression equations incorporating latent variable and semiparametric modeling techniques. Our choice approach to modeling nonlinear relationship is the use of the Additive Model (AM) technique (Hastie & Tibshirani, 1990). This method employs smooth functions which (in a flexible way) allow data to define the relationships between response and predictor variables. For example, in Kellam, et al.,(2008) and Brown, et al., (2008) the Additive modeling technique was used to examine intervention impact on certain distal outcomes (e.g. drug abuse/dependence) in terms of smooth functions of the observed aggressive behavior at baseline and of the baseline-treatment interaction term. Specifically, the

authors employed the Generalized Additive Model (GAM) technique (Hastie & Tibshirani, 1990) for the binary outcomes. The theory of intervention that guided their work posited that the intervention would succeed to the extent that it does affect the risk factor. If someone is already at the lowest level of risk, i.e., does not exhibit aggressive behavior, then the impact of the intervention is expected to be low or nonexistent.

In the context of baseline measures, two notable statistical issues commonly arise in social and psychological research, and how they are addressed may have serious implications to validity of statistical inference. The first situation is when baseline risk factors are not directly observed and have to be estimated from multiple observed indicators before each can be used to predict any outcome in a regression model. Second, a nonlinear relationship between the baseline and the outcome is often closer to reality than linear relationship. One analytic approach to resolve these issues is to combine two existing methodologies sequentially in a complimentary way: latent variable (LV) modeling and additive modeling (AM). Essentially one can perform a two-step analysis where the Empirical Bayes' (EB) estimate (or factor scores) of the baseline risk is first obtained for individuals via a Confirmatory Factor Analysis (CFA), then follow up with a nonlinear regression analysis such as the GAM (Hastie & Tibshirani, 1990) that includes the baseline estimate as a predictor (Toyinbo, P. A., & Brown, C. H. 'Variation in Intervention Impact in Adolescent Substance Abuse Prevention Study'. *Report presented on April 18, 2007 at the Adolescent Substance Abuse Prevention Study National Advisory Group Meeting, Institute for Health and Social Policy, University of Akron* . Akron, Ohio, USA.). The approach adopted here led to the novel concept of Additive Latent Variable (ALV) modeling technique (the original name was coined by the second author).

We noted that a strong feature of the SEM framework is the capacity for evaluating the whole model as a unit with several available and useful model comparison and goodness-of-fit tests. However when a two-step analysis is carried out as described above, the simultaneous solution advantage is lost. In addition such an ad-hoc approach suffers from the usual estimation errors of factor scores (Muthen, 1989). Also, as at the time of this study, the two-step analysis requires different statistical software platforms for the different implementation steps, hence the method is not efficient. Ideally, a simultaneous solution to the system of equations consisting of both LV and GAM models will be more efficient but, to the best of our knowledge, such method has never been reported in the literature. We hope to begin to fill this gap in our current study. Based on the idea of bringing together several different existing analysis types in one general model under a unifying framework of the SEM (Muthen, 2002), an attractive approach to filling the gap we have identified is to integrate GAM into the very powerful and flexible SEM framework to allow for simultaneous solutions to one system of equations. This will allow maximal strengths to be drawn from both component modeling techniques.

1.3 Proposing a New Methodology: Additive Latent Variable Model

This dissertation concerns the development of a new method for analyzing variation in impact of a universal preventive intervention. The new integrated ALV modeling technique we are proposing here consists of two existing analytic techniques (LV and GAM) and will allow for the evaluation of the combined analyses as a whole unit. By integrating additive models into a LV framework, a single system of equations can be solved simultaneously, and many of the powerful tests possible in LV modeling may be available globally. This new method has a great potential for extensions and will provide an opportunity to examine smoothed nonlinear relationships between latent variable constructs (such as baseline risk, growth factors) and a proximal or distal

outcome, within the LV framework. The method will also examine how these relationships interact with treatment. What is unique about the proposed model is that it incorporates two powerful analytic techniques into one modeling technique which to date has never been reported. Moreover, the ALV will allow for a more efficient use of the data, alleviate the problem of multiple comparison tests as well as be ridden of the need for multiple statistical application platforms. The ALV modeling technique can be used to analyze cross-sectional or longitudinal data. In the longitudinal case it allows us to examine how the effect of treatment on a long term outcome may interact with the mediating growth process, particularly when such interaction is non-linear.

This dissertation is motivated by analysis requirements of the Adolescent Substance Abuse Prevention Study (ASAPS) data (Sloboda, et al., 2008). The ASAPS study, funded by the Robert Wood Johnson Foundation and conducted by the Institute for Health and Social Policy of the University of Akron, is a cluster randomized field trial of 83 school clusters consisting of high school and all its feeder middle schools with a total of 19,200 students from six metropolitan areas spread across the country. The intervention program: Take Charge of Your Life (TCYL) was delivered to students in the 7th grade and repeated in the 9th grade. Seven self-administered survey waves were administered, starting from 7th grade (pre-intervention) to the 11th grade, to collect data on behavioral outcomes including use of alcohol, tobacco and other drugs.

A major research question confronting the researchers is “who benefits or is harmed by this intervention?” To put it in another way, does the intervention affect individuals differently as a function of baseline risk? To answer this major research question the analyses will benefit immensely from both the Latent Variable and Additive modeling techniques. The Additive model which uses a smoother to summarize the potential nonlinear trends in the data complements the

Latent Variable model. The Latent Variable modeling technique can be used to summarize the observed baseline risk indicators into a one-dimensional 'error-free' latent baseline risk, from which the EB estimates can be computed. The GAM model part then describes the dependence (linear or nonlinear) of the outcome on baseline risk and the baseline interaction with treatment (see also Brown, 1993). So the GAM method can help distinguish whether the effect of the intervention is similar across all levels of risk (or other baseline covariates of interest), limited to low or high risk, or beneficial for some but harmful for others. Furthermore, for hierarchical data such as the ASAPS data, the GAM can be substituted with a generalized additive mixed model (GAMM) counterpart to add school level random effects into the nonlinear regression sub-model. A major limitation to the two-step modeling approach described above is that the two analytic methods required two different statistical application platforms for implementation, so the approach is not efficient. The purpose of this dissertation is to address the deficiency by integrating the two components into one single model that performs simultaneous solutions.

In the next chapter, we will discuss the basic statistical theory for the development and estimation of the proposed integrated ALV model. First we describe the SEM as a standard template for LV models. The EM algorithm will be set up for the maximum likelihood estimation such that it will allow for a later incorporation of the additive model to into the LV framework. The Markov Chain Monte Carlo fitting method for handling resultant complex integrations is also presented here. The remainder of the dissertation is organized as follows. Chapter 3 is devoted to the basic theory and estimation of Additive Models. Here the GAM is finally integrated into the LV framework which employs the EM algorithm for estimation as set up in chapter 2. Chapter 4 is concerned with the computational development of the ALV model algorithm while detailed simulation study of the performance of the ALV estimation method is presented in chapter 5. In

chapter 6 the new ALV modeling technique is applied to a portion of the ASAP data, and finally the discussion is presented in chapter 7.

CHAPTER 2

FRAMEWORK FOR ALV MODEL FORMULATION

2.1 General SEM Framework

The structural equation modeling (SEM) is a powerful and very flexible analytic tool. To illustrate how expandable the SEM modeling framework is, we describe here a general model where continuous latent variables represent some unobservable constructs of substantive significance, but are indirectly measured by multiple observable variables or factor indicators, and are also influenced by covariates. This model type represents a member of the SEM family, the general form of which can be specified in two parts: measurement and structural sub-models (Muthen, 2002).

The measurement sub-model of the SEM links a $m \times 1$ vector of latent variables η to a $p \times 1$ vector of factor indicator variables observed on *i*th unit, as follows:

$$\mathbf{y_i} = \upsilon + \Lambda\eta_i + \mathrm{K}\mathbf{x_i} + \varepsilon_i \tag{2.1}$$

where υ is a $p \times 1$ parameter vector of measurement intercepts, x is a $q \times 1$ vector of covariates, ε is a $p \times 1$ vector of measurement errors or residuals with a $p \times p$ covariance matrix denoted Θ, Λ is a $p \times m$ parameter matrix of factor loadings or measurement slopes. K is a $p \times q$ parameter matrix of regression slopes measuring the direct influence on the outcome vector y by covariates x. For some applications, the flexibility of the SEM general framework is exploited to allow for the measurement intercept vector υ to be used for multiple groups modeling or for growth

modeling with multiple indicators at each time point. The structural sub-model is specified as follows:

$$\boldsymbol{\eta}_i = \boldsymbol{\alpha} + \mathbf{B}\boldsymbol{\eta}_i + \boldsymbol{\Gamma}\mathbf{x}_i + \boldsymbol{\zeta}_i \tag{2.2}$$

where $\boldsymbol{\alpha}$ is an $m \times 1$ parameter vector of structural intercepts, $\boldsymbol{\Gamma}$ is an $m \times q$ parameter matrix of slopes where latent variables are regressed on covariates, and $\boldsymbol{\zeta}$ is an $m \times 1$ vector of structural errors and has a $m \times m$ covariance matrix $\boldsymbol{\Psi}$. This formulation is written with the latent variables on both sides of the equation to allow for some latent variables to be predicted by others or one another (simultaneous equations with recursive or non-recursive relationships). So $\boldsymbol{B}$ is a restricted, $m \times m$ matrix of regression coefficients that relate latent variables to one another (either recursively or non-recursively). We will limit our focus in this study to recursive models (no feedback loops); that is, $\boldsymbol{B}$ matrix is restricted to lower triangular with zeros in its diagonal so that the independent latent variables do not co-vary with dependent latent variables.

The above formulations of the SEM are equivalent to the original LISREL model (Joreskog, 1977) with the usual model assumptions as follows. The measurement errors $\boldsymbol{\varepsilon}$'s and structural errors $\boldsymbol{\zeta}$'s are mutually independent among the units, and both vectors are not correlated. We apply the standard assumptions about the error distribution. The $\boldsymbol{\varepsilon}$'s and $\boldsymbol{\zeta}$'s are assumed to have a multivariate normal distribution with zero expectations. Also the $\boldsymbol{\zeta}$'s are independent of the exogenous latent variables.

$$\begin{aligned}
&\boldsymbol{\varepsilon} \sim N_p(0, \boldsymbol{\Theta}) \\
&\boldsymbol{\zeta} \sim N_m(0, \boldsymbol{\Psi}) \\
&\mathrm{Cov}(\boldsymbol{\varepsilon}, \boldsymbol{\zeta}) = 0
\end{aligned}$$

These assumptions imply that the latent independent variables are multivariate normal. The independent covariates that are measured without error may have any distribution since we will be conducting our analysis conditional on these covariates. The general formulation for SEM above can be re-expressed in a more traditional regression framework by solving for $\boldsymbol{\eta}$ and then pre-multiplying by $(\boldsymbol{I} - \boldsymbol{B})^{-1}$:

$$\boldsymbol{\eta} = (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\alpha} + (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma}\mathbf{x} + (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\zeta} \tag{2.3}$$

$$\mathbf{y} = \boldsymbol{\upsilon} + \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\alpha} + \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma}\mathbf{x} + \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\zeta} + \mathbf{K}\mathbf{x} + \boldsymbol{\varepsilon} \tag{2.4}$$

We prefer to work with this reduced model form. Thus for the general SEM, the mean and covariance structures conditional on $\boldsymbol{x}$ are respectively

$$\boldsymbol{\upsilon} + \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\alpha} + \boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma}\mathbf{x} + \mathbf{K}\mathbf{x} \tag{2.5}$$

$$\boldsymbol{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Psi}(\mathbf{I} - \mathbf{B})'^{-1}\boldsymbol{\Lambda}' + \boldsymbol{\Theta} \tag{2.6}$$

with the restriction that $\boldsymbol{I} - \boldsymbol{B}$ is non-singular.

For convenience, we will consider a simple and less general member of the SEM family where $\boldsymbol{y}$ is a vector of indicators of a single latent variable and there are no covariates. This simple model can be easily specified by introducing different constraints into the general SEM model framework (2.2 and 2.3). With no covariates $\boldsymbol{x}$ in the model, $\boldsymbol{K}$ and $\boldsymbol{\Gamma}$ diminish. Such is the confirmatory factor analytic (CFA) model which is a system of linear regressions where each of the observed variables is predicted by one or more latent variables based on theory. A conventional CFA model is a simple recursive model with additional constraints that include

$\boldsymbol{B} = 0$ in (2.3). This model can be represented by the following simplified results from (2.3) and (2.4) respectively:

$$\mathbf{y} = \upsilon + \Lambda\eta + \varepsilon \tag{2.7}$$

$$\eta = \alpha + \zeta \tag{2.8}$$

$$\mathbf{y} \sim \mathrm{N}_{\mathrm{p}}(\upsilon, \Lambda\Psi\Lambda' + \Theta); \quad [\mathbf{y} \mid \eta] \sim \mathrm{N}_{\mathrm{p}}(\upsilon + \Lambda\eta, \Theta) \tag{2.9}$$

In this model, the loadings are the regression coefficients of the latent variables. This would be a traditional multivariate regression model (with restricted regression coefficients) if the latent variables were observed. With latent variables being unobservable there is a built in indeterminacy where affine transformations of the latent variables can yield the same fit but completely different loadings and other parameters. To resolve this identifiability issue, sufficient number of restrictions is placed on either Λ (one element in each column is fixed at one) or on Ψ (diagonal elements set to one); and also on $\boldsymbol{\alpha}$. A convenient way to remove this model indeterminacy is to fix the means of $\boldsymbol{\eta}$ to zero and the covariance to the identity. It means that the factors are orthogonal and the factor components are independent under normal distribution. Using this specification, the mean vector $\boldsymbol{\upsilon}$ is unrestricted and therefore optimally estimated (under ML and other such models) by the observed sample means for $\boldsymbol{y}$. The vector of the parameters to be estimated is $\boldsymbol{\Omega}_{\boldsymbol{y}} = (\boldsymbol{\upsilon}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}) = (\upsilon_1,...,\upsilon_p,\lambda_1,...,\lambda_p,\theta_1,...,\theta_p)$.

Assuming a standard normal marginal distribution for $\boldsymbol{\eta}$, define a p-variate vector of variables $\boldsymbol{Y}_i$ observed on the *i*th subject, i=1,…,N ; assume that the *N* observations are independent and each vector $\boldsymbol{Y}_i$ conditionally has multivariate normal density written as

$$
\begin{aligned}
f(\mathbf{y_i} \mid \boldsymbol{\eta_i}; \boldsymbol{\Omega_y}) &= (2\pi)^{-\mathbf{p}/2} \left|\boldsymbol{\Theta}\right|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{y_i} - \boldsymbol{\upsilon} - \boldsymbol{\Lambda}\boldsymbol{\eta_i})'\boldsymbol{\Theta}^{-1}(\mathbf{y_i} - \boldsymbol{\upsilon} - \boldsymbol{\Lambda}\boldsymbol{\eta_i})\right], \\
f(\boldsymbol{\eta_i}) &= (2\pi)^{-\mathbf{m}/2} \exp\left[-\frac{1}{2}\boldsymbol{\eta_i}'\boldsymbol{\eta_i}\right];
\end{aligned}
\tag{2.10}
$$

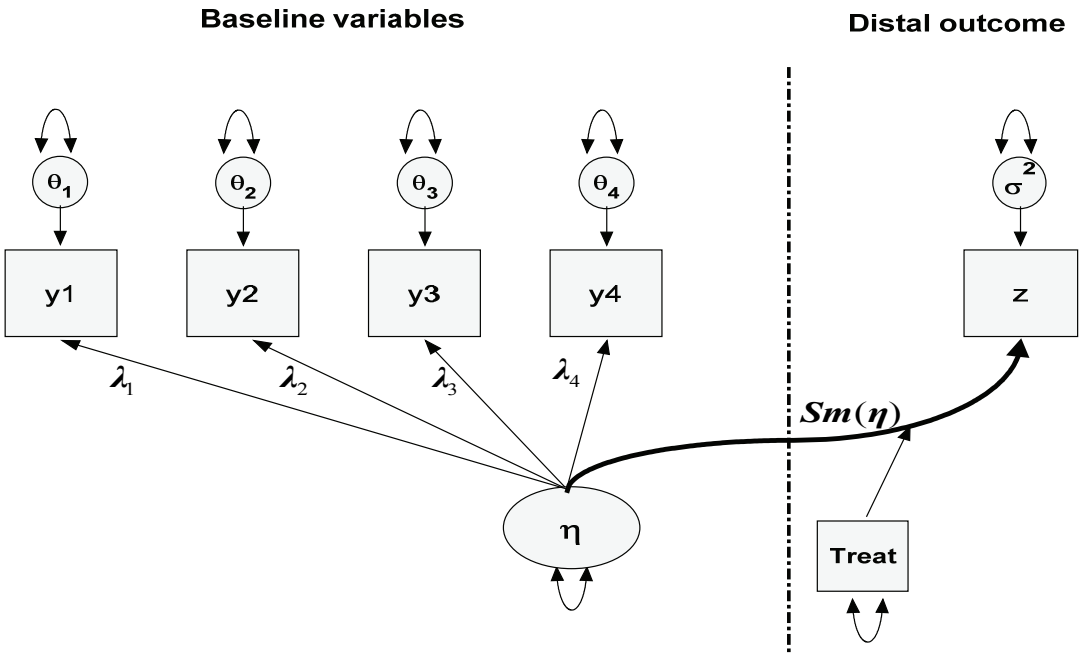
where $\boldsymbol{\upsilon}$ is the mean vector of $\boldsymbol{y_i}$. Next we will introduce the ALV model building blocks and the algorithms to be used to optimize the model parameters. Implementation details will be addressed in subsequent chapters.

2.2 Basic Formulation of ALV Model

In the ASAPS study, each of *N* individuals in the ASAPS study is associated with a set of observations including the baseline risk variables, baseline covariates, distal outcome and intervention status data. Let the vector of risk measures be denoted by $\boldsymbol{Y} = \{Y_1, ..., Y_p\}$ which serve as indicators of a single underlying latent risk factor denoted by η. We wish to examine how the unobserved baseline risk factor may predict the observed distal outcome Z given an intervention status *Treat* and fixed covariates *X*. Specifically we are interested in modeling the potential smooth non-linear relationship between Z and η.

With little modifications to the model diagram conventions adopted in (Muthen & Muthen, 2008) , the path diagram for our proposed ALV model when p = 4 is depicted in the Figure 2.1. The straight lines with single arrow head represent linear regressions where the $\boldsymbol{\lambda}$'s are factor loadings, that is regression coefficients of the **Y**'s on the latent variable η. The thick curve counterpart represents a potential non-linear relationship to be captured by the regression of Z on η which may appropriately require anything from non-linear parametric to non-parametric modeling approach, such as the use of a smooth function (*Sm*) in the ALV model.

Figure 2.1 The Proposed ALV Model Diagram



The proposed ALV model therefore involves the integration of Generalized Additive Model (GAM) into a LV model framework. Let the LV model be defined as a simple latent factor model of the observed data $\{\boldsymbol{y}_i, z_i\}$ without covariates and be represented by the probability density function

$$p(\mathbf{y}_i, z_i|\eta_i;\boldsymbol{\Omega})$$

where $\boldsymbol{\Omega}$ is the updated set of all parameters to be estimated. For now we treat the population as a single homogenous group and there is no additional explanatory variable to be considered in the model. That means we have an $N \times (p+1)$ observed data matrix consisting of p factor indicators

plus a single distal outcome. We assume $\boldsymbol{Y_i}$ are continuous and are linearly related to an underlying latent factor η_i, that is, $\boldsymbol{Y_i}$ retains the conventional linear factor analysis model structure and the latent variable model is identifiable. In preparation for the new work on this dissertation, we do not require the response Z_i to be normally distributed or to be linearly related to η_i, but we will always assume that $\boldsymbol{Y_i}$ and Z_i are independent given η_i. With $\boldsymbol{Y_i}$ and Z_i conditionally independent, we can decompose the LV model into two independent joint conditional likelihood functions:

$$\begin{aligned} L(\boldsymbol{\Omega}; \mathbf{y}, z|\boldsymbol{\eta}) &= \prod_{i=1}^{N} p(\mathbf{y_i}, \mathbf{z_i} \mid \eta_i; \boldsymbol{\Omega}) \\ &= \prod_{i=1}^{N} p(\mathbf{y_i} \mid \eta_i; \boldsymbol{\Omega}_y) \times p(z_i \mid \eta_i; \boldsymbol{\Omega}_z) \end{aligned} \tag{2.11}$$

where $\boldsymbol{\Omega}_y$ and $\boldsymbol{\Omega}_z$ are distinct set of parameters in $\boldsymbol{\Omega}$. The first component of the LV model then represents a simple confirmatory factor analysis (CFA) model consisting of a system of *p* linear regression equations while the second component is simply a regression model of the distal outcome on the latent factor. To convert the LV into ALV we simply represent the second component as a GAM of a distal outcome Z where Z can be a continuous, categorical or count variable.

2.3 ML Estimation of ALV Model via the EM Algorithm

We adopt a likelihood based approach to parameter estimation. It is anticipated that eventually when we fully specify the two component density functions constituting the ALV model, performing direct maximization of the observed data likelihood will be complicated by the presence of non-linear relationship between the variables and the associated parameters and the intractable integrals that may result. Therefore we choose to implement maximum likelihood

estimation using the Expectation-Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). The EM algorithm is a general and easily adaptable approach for finding the maximum likelihood estimates (mle's) of the underlying parameters in a given data when the data are incomplete or have missing values. In our case the observed data $\{\boldsymbol{y}, \mathrm{z}\}$ depend on a latent factor η which is unobservable. So we consider the situation as a missing data problem, where η is treated as missing at random (Rubin, 1976). Our specifications of the EM algorithm will be based on regarding η as a random N-vector of missing data within the SEM model framework. We then treat the observed $\{\boldsymbol{y}, \mathrm{z}\}$ as incomplete data while $\{\boldsymbol{y}, \mathrm{z}, \eta\}$ constitute complete data in which the rows are independently and identically distributed (Dempster, Laird, & Rubin, 1977). We will develop an EM procedure for parameter estimation that allows for a non-linear regression of Z on the latent factor η via a smooth function.

The complete-data likelihood is expressed as

$$\begin{aligned} \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega}) &= \mathrm{p}(\mathbf{y}, \mathrm{z}, \eta; \boldsymbol{\Omega}) \\ &= \mathrm{p}(\mathbf{y} \mid \eta; \boldsymbol{\Omega}_{\mathrm{y}}) \times \mathrm{p}(\mathrm{z} \mid \eta; \boldsymbol{\Omega}_{\mathrm{z}}) \times \mathrm{p}(\eta; \boldsymbol{\Omega}) \end{aligned} \tag{2.12}$$

where the random η is unknown and, given a factor analytic model framework, its marginal distribution is fixed as standard normal for model identification purpose (see equation (2.10)). The maximum likelihood estimates of Ω will be computed from the complete data with the above specifications and restrictions.

Consider that if η were observed, then we have a simple distribution for the 'complete' data where mle's for $\boldsymbol{\Omega}$ can be obtained by the usual least square method based on the sums, sums of squares and sums of cross-products (Dempster, Laird, & Rubin, 1977). Normally, when η is not observed, we would obtain the mle's of the parameters by integrating the complete-data

likelihood $\boldsymbol{p}(\boldsymbol{y}, z, \eta; \boldsymbol{\Omega})$ with respect to η and maximizing the results with respect to $\boldsymbol{\Omega}$. However the approach to estimation taken by the EM algorithm is to alternate between computation of the expectation of the complete data log-likelihood (E-step) and the maximization of this expectation with respect to $\boldsymbol{\Omega}$ (M-step). The idea is to fill in a set of values for the 'missing' η (E-step) and solve the problem, i.e. find mle's for $\boldsymbol{\Omega}$ (M-step); the repeat the two steps to find better values of η to fill in (Rubin,1991). Because η is unknown, draws from its conditional distribution $p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega})$ will be taken to simulate η. Let $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ be defined as the *k*th iterative expected complete data log-likelihood given the observed data and current values of $\boldsymbol{\Omega}^{(k)}$, which is given by

$$\begin{aligned} Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(\mathbf{k})}) &= E^{\eta}[\log L_{com}(\boldsymbol{\Omega}) \mid \mathbf{y}, z, \boldsymbol{\Omega}^{(\mathbf{k})}] \\ &= E^{\eta}[\log p(\mathbf{y}, z, \eta; \boldsymbol{\Omega}) \mid \mathbf{y}, z, \boldsymbol{\Omega}^{(\mathbf{k})}] \end{aligned} \tag{2.13}$$

Each repetition of the two steps yields a new set of mle's for $\boldsymbol{\Omega}$ by numerically increasing the value of quantity $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ and the iteration continues until convergence. One important property of the EM algorithm is that a (*k+1*)th iteration causes $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ to increase over its *k*th value (Dempster, Laird, & Rubin, 1977).

Briefly, $\boldsymbol{\Omega}$ contains the parameters to be optimized as $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ increases; with a current value $\boldsymbol{\Omega}^{(k)}$, iteration *k* of the EM Algorithm is implemented in the following sequences:

1. Draw from the conditional distribution $p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)})$, (i.e. evaluate it at the current parameter estimates $\boldsymbol{\Omega}^{(k)}$); supply initial values if k=0.

2. E-step: Evaluate $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ using updates from (1), that is taking the expectation of the complete data log-likelihood with respect to the conditional distribution $p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)})$.

3. M-step: Maximize $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ over $\boldsymbol{\Omega}$ to obtain a revised $\boldsymbol{\Omega}^{(k+1)}$. That is, solve

$$\boldsymbol{\Omega}^{(k+1)} = \max_{\Omega} Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$$

4. Check for convergence, if none, set $\boldsymbol{\Omega}^{(k)} = \boldsymbol{\Omega}^{(k+1)}$ and return to (1).

2.3.1 The Expectation Step of EM

The E-step at the *k*th iteration computes the expected value of the complete data log-likelihood over η given the observed data and current values of $\boldsymbol{\Omega}^{(k)}$. This step is more formally defined as

$$\begin{aligned} Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(\mathbf{k})}) &= \int \log p(\mathbf{y}, z, \eta; \boldsymbol{\Omega}) \times p(\eta; \boldsymbol{\Omega}^{(\mathbf{k})}) d\eta \\ &\propto \int \log p(\mathbf{y}, z, \eta; \boldsymbol{\Omega}) \times p(\eta \mid \mathbf{y}, z; \boldsymbol{\Omega}^{(\mathbf{k})}) d\eta \end{aligned} \tag{2.14}$$

where the complete data likelihood derives its randomness solely from being a function of random variable η that is governed by its conditional predictive distribution given the observed data: $f(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)})$. The complete data log-likelihood function is not tractable analytically because we do not have a fully known parametric form for the joint distribution $p(\boldsymbol{y}, z, \eta; \boldsymbol{\Omega})$, therefore we require an alternative method to direct integration in the E-step.

Markov Chain Monte Carlo (MCMC) Method

Whenever the computations involved in the integration (E-step) and /or optimization (M-step) are intractable, numerical methods or Monte Carlo methods may be indicated (Wei &

Tanner, 1990; McLachlan & Krishnan, 2008). Our choice here, the Monte Carlo method, computes integrals using random number generation, and it is preferred to numerical quadrature methods when the dimension of integral may be large or the functions may not be smooth (McLachlan & Krishnan, 2008). For simplicity we illustrate with an example of a complex integral $I(y) = \int f(y \mid x)p(x)dx$ which can be expressed as an expectation of $f(y \mid x)$ over the continuous density $p(x)$. To use the classical Monte Carlo integration (McLachlan & Krishnan, 2008; Walsh, 2004), a sufficiently large number $x_1, .., x_c, .., x_C$ of random sample are drawn from the density $p(x)$ (which must be completely known) and the integral is approximated by

$$\hat{I}(y) = \frac{1}{C}\sum_{c=1}^{C} f(y \mid x_c) \tag{2.15}$$

The estimated variance of the Monte Carlo estimate is given by

$$\hat{var}[\hat{I}(y)] = \frac{1}{C}\left(\frac{1}{C-1}\sum_{c=1}^{C}\left(f(y \mid x_c) - \hat{I}(y)\right)^2 \right)$$

If the target distribution $p(x)$ itself is complex and is indirectly or incompletely specified, then a more complex Monte Carlo method will be required (McLachlan & Krishnan, 2008). For example when $p(x)$ is uniquely defined but does not have a standard form that is amenable to direct sampling, Markov chain Monte Carlo (MCMC) methods are commonly used to draw samples indirectly from these distributions (McLachlan & Krishnan, 2008; Lee & Song, 2007; Wei & Tanner, 1990).

A Markov chain is a stochastic process that characterizes sequences of random variables, where "the transition probabilities between different values in the state space depend only on the random variable's current state" (Walsh, 2004; Gamerman & Lopes, 2006). The most critical

feature that defines a particular Markov chain is the transition kernel (transition probabilities) which is the limiting distribution of the chain. The aim therefore is to construct a Markov chain such that its limiting distribution equals the target distribution we wish to simulate.

Let the transition kernel be defined as $q(x^{(c)}, x^{(c+1)})$ which is the probability of transition of a process from an earlier state $x^{(c)}$ to the next state $x^{(c+1)}$ in a single step (Walsh, 2004). To draw a random sample from distribution $p(x)$ via Markov chains, the transition kernel must be chosen such that the stationary distribution of the chain is $p(x)$ (Gamerman & Lopes, 2006), and $q(,)$ must satisfy

$$\begin{aligned} & p(x^{(c)})q(x^{(c)}, x^{(c+1)}) = p(x^{(c+1)})q(x^{(c+1)}, x^{(c)}), \quad \forall (x^{(c)}, x^{(c+1)}) \\ & \text{i.e.} \quad \frac{p(x^{(c)})}{p(x^{(c+1)})} = \frac{q(x^{(c+1)}, x^{(c)})}{q(x^{(c)}, x^{(c+1)})}. \end{aligned} \tag{2.16}$$

This is the basis for the Metropolis-Hastings Algorithm which we will discuss next.

Metropolis-Hastings Algorithm

The Metropolis-Hastings (M-H) Algorithm is a very widely applicable MCMC method for simulating a complex nonstandard multivariate distribution; the Gibbs sampler (Geman & Geman, 1984) is a special case of the M-H algorithm (Walsh, 2004). The mechanism of the M-H algorithm as outlined in Gamerman & Lopes (2006) and Walsh (2004) will be described briefly here. Note from the q ratio above that it is sufficient to be able to express a qualifying stationary distribution $p(x)$ up to the normalizing constant, since any constant factor cancels out when calculating the transition kernel. Suppose we wish to draw samples from $p(x): x = (x_1, ..., x_d)'$ of which direct sampling is complicated. If $f(x)$ is an approximation up to a constant and is

available, such that $p(x) = f(x)/h$ where the normalizing constant *h* is difficult to compute, we can generate a d-dimensional random vector from $f(x)$ using the M-H algorithm. For the M-H scheme, first a proposal kernel (density) $q(x^{(c)}, x^{(c+1)})$ is chosen so as to be as similar to the target density $p(x)$ as possible, to increase acceptance rate. Note that if the sampling (proposal) kernel equals the target distribution (i.e. when the latter is known), then acceptance rate is 100 percent and direct draw from the target density itself is possible, as in the classical Monte Carlo procedure. Desirable features of a proposal kernel include tunable parameters such as location and scale (Chib & Greenberg, 1995; Walsh, 2004). A widely used proposal kernel (or candidate-generating density) is the multivariate normal.

The following steps are carried out in the M-H scheme: (I) choose arbitrary initial values x_0 satisfying $f(x_0) > 0$, (II) evaluate the proposal (or jumping) distribution $q(x^{(c)}, x^{(c+1)})$ at the current x_0 values, and then sample a candidate point x^* from $q(x^{(c)}, x^{(c+1)})$, (III) define an acceptance probability of a move of the chain from current value $x^{(c)}$ to a new value $x^{(c+1)}$ as the ratio of the densities at the proposal point x^* and current point $x^{(c)}$:

$$\alpha = \frac{p(x^*)}{p(x^c)} = \frac{f(x^*)/h}{f(x^c)/h} = \frac{f(x^*)}{f(x^c)} \tag{2.17}$$

If $\alpha \geq 1$, a move to the new proposal point increases the density and so is allowed, else the move is allowed with a probability of α. The basis for allowable move can be summarized as

$$\alpha(x^c, x^*) = \min\left\{1, \frac{f(x^*)q(x^c, x^*)}{f(x^c)q(x^*, x^c)}\right\}. \tag{2.18}$$

(IV) To introduce randomness a quantity u is generated from an independent uniform distribution $U(0,1)$, then the proposal point is accepted as the current value $x^* = x^{(c+1)}$ if $\alpha \geq u$, else it is rejected and no change takes place, i.e. $x^* = x^{(c)}$. Steps II to IV are iterated until convergence. The above Metropolis-Hastings algorithm is a generalization of the original Metropolis algorithm (McLachlan & Krishnan, 2008; Walsh, 2004). The original algorithm requires that the proposal density be symmetric (e.g. normal distributions): $q\left(x^c, x^*\right) = q\left(x^*, x^c\right)$ so that $\alpha(x^c, x^*)$ reduces to

$$\alpha(x^c, x^*) = \min\left\{1, \frac{f(x^*)}{f(x^c)}\right\} \tag{2.19}$$

Expectation of Complete Data Log-likelihood

To reiterate we are adopting a method similar to the Monte Carlo EM algorithm (MCEM) (Wei & Tanner, 1990) whereby the Monte Carlo integration of the log-likelihood is approximated by drawing a sufficiently large number C of observations from the predictive conditional distribution $p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)})$ evaluated at the current values $\boldsymbol{\Omega}^{(k)}$. Upon generating the random observations $\{\eta_i^{(c)}, c=1,...,C, i=1,...,N\}$ by the MH algorithm, there are different ways to use the observations in both the E-step and M-step. For the E-step, a popular and straight forward process is to plug the expected value of the Markov process generated random observations (or its function of some sufficient statistics) directly into the $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ function (Lee & Zhu, 2000). For another example, these random observations were plugged into conditional expectations of the complete data approximate sufficient statistics in (Lee & Zhu, 2002) to evaluate the E-step. In our case, in the E-step we decided to fill in the entire estimated density of $p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)})$ into $Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)})$ so the problem considerably simplifies to that of a C number of simple regression

equations with fixed covariates, similar to an example described in McLachlan & Krishnan (2008). Note that $\boldsymbol{\Omega}^{(k)}$ is already imbedded in the C drawings:

$$\hat{Q}(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)}) = \frac{1}{C}\sum_{c=1}^{C}[\log p(\mathbf{y}, z, \eta^{(c)}; \boldsymbol{\Omega})] \tag{2.20}$$

A single scan or generated sequence $\eta_i^{(-t)},...,\eta_i^{(0)},\eta_i^{(1)},...,\eta_i^{(c)},...,\eta_i^{(C)}$ is a Markov chain for the *i*th subject. As *c* tends to infinity, or with a sufficiently large C, the stationary distribution converges in distribution to the target distribution $p(\eta_i \mid \boldsymbol{y_i}, z_i; \boldsymbol{\Omega}^{(k)})$. To allow a sufficient amount of time for a stationary distribution to be reached, the first set of iterations in the chain $\eta_i^{(-t)},...,\eta_i^{(0)}$ serves as the burn-in segment. This initial set of iterations is discarded while the remainder segment of the chain forms the sample of an optimal finite size C to be used in the Monte Carlo integration. The usable Markov sample then consists of limiting transition probabilities that are no longer dependent on the start values. However, by using successive values from a single Markov chain per subject, within-subject autocorrelation does induce chain dependence. In order for inference based on the sample to still be valid, higher autocorrelation will require a longer chain to run (Gamerman & Lopes, 2006). The authors also noted that Markov chains only have first order dependence which decreases with increasing lag between iterations, therefore subsample of quasi-independent elements can be formed by storing only every *j*th value post burn-in period. This method is referred to as 'thinning' and it also has the advantage of requiring relatively shorter chains. With thinning we can achieve independence in the final sample with improved optimality and, in addition, reduce storage requirement for computer generated data. Furthermore, by generating a Markov sample independently for each subject, we are able to make the assumption of both within-subject and between-subject

independence for the N Markov samples. Therefore by drawing a sufficiently large sample $\eta_i^{(c)}$,c=1,...,C from $p(\eta \mid \mathbf{y}, z; \boldsymbol{\Omega}^{(k)})$ we can write

$$\hat{Q}(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)}) = \frac{1}{C} \sum_{c=1}^{C} \sum_{i=1}^{N} [\log p(\mathbf{y_i}, z_i, \eta_i^{(c)}; \boldsymbol{\Omega})] \tag{2.21}$$

Since we are using Metropolis algorithm to sample from a conditional normally distributed η, the selection probability simplifies to $p(\eta^* \mid \boldsymbol{y}, z; \boldsymbol{\Omega}) / p(\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega})$ where η^* is the proposal value. Recall that given η, $\boldsymbol{Y}$ and Z are independent. Therefore for the *i*th subject in the *k*th EM iterate the conditional distribution can be approximated up to a constant *K* as follows:

$$\begin{aligned} p(\eta_i \mid \mathbf{y_i}, z_i; \boldsymbol{\Omega}^{(k)}) &= \frac{p(\eta_i, \mathbf{y_i}, z_i; \boldsymbol{\Omega}^{(k)})}{p(\mathbf{y_i}, z_i; \boldsymbol{\Omega}^{(k)})} = \frac{1}{h} \times p(\eta_i, \mathbf{y_i}, z_i; \boldsymbol{\Omega}^{(k)}); \\ p(\eta_i \mid \mathbf{y_i}, z_i; \boldsymbol{\Omega}^{(k)}) &= \frac{1}{h} \times p(\mathbf{y_i} \mid \eta_i; \boldsymbol{\Omega}_y^{(k)}) p(z_i \mid \eta_i; \boldsymbol{\Omega}_z^{(k)}) p(\eta_i). \end{aligned} \tag{2.22}$$

So we have (for normal Z linearly related to η)

$$\begin{aligned} f(\mathbf{y_i} \mid \eta_i; \boldsymbol{\Omega}_y) &= (2\pi)^{-p/2} \left|\boldsymbol{\Theta}\right|^{-1/2} \exp\left[-\frac{1}{2} (\mathbf{y_i} - \boldsymbol{\upsilon} - \boldsymbol{\Lambda}\boldsymbol{\eta_i})' \boldsymbol{\Theta}^{-1} (\mathbf{y_i} - \boldsymbol{\upsilon} - \boldsymbol{\Lambda}\boldsymbol{\eta_i}) \right], \\ f(z_i \mid \eta_i; \boldsymbol{\Omega}_z) &= (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{1}{2\sigma^2} (z_i - a - \beta\eta_i)^2 \right], \\ f(\eta_i) &= (2\pi)^{-m/2} \exp\left[-\frac{1}{2} \eta_i' \eta_i \right]. \end{aligned} \tag{2.23}$$

In the *c*th MCMC iteration the candidate value η_i^* drawn from the univariate normal proposal distribution is accepted as the new value $\eta_i^{(c+1)}$ with the probability of α:

$$\alpha(\eta_i^{(c)}, \eta_i^*) = \min\left\{ 1, \quad \frac{p(\mathbf{y_i} \mid \eta_i^*; \boldsymbol{\Omega}_y^{(k)}) p(z_i \mid \eta_i^*; \boldsymbol{\Omega}_z^{(k)}) p(\eta_i^*)}{p(\mathbf{y_i} \mid \eta_i^{(c)}; \boldsymbol{\Omega}_y^{(k)}) p(z_i \mid \eta_i^{(c)}; \boldsymbol{\Omega}_z^{(k)}) p(\eta_i^{(c)})} \right\} \tag{2.24}$$

Note that *h* has cancelled out in the ratio $\alpha(\eta_i^{(c)},\eta_i^*)$. From (2.23) the ratio therefore simplifies to

$$\min\left\{1, \frac{\exp\left[-\frac{1}{2}\left\{(\mathbf{y_i}-\boldsymbol{\upsilon}-\boldsymbol{\Lambda}\eta_i^*)'\boldsymbol{\Theta}^{-1}(\mathbf{y_i}-\boldsymbol{\upsilon}-\boldsymbol{\Lambda}\eta_i^*)+\sigma^{-2}(z_i\text{-}a\text{-}b\eta_i^*)^2+\eta_i^{*\prime}\eta_i^*\right\}\right]}{\exp\left[-\frac{1}{2}\left\{(\mathbf{y_i}-\boldsymbol{\upsilon}-\boldsymbol{\Lambda}\eta_i^{(c)})'\boldsymbol{\Theta}^{-1}(\mathbf{y_i}-\boldsymbol{\upsilon}-\boldsymbol{\Lambda}\eta_i^{(c)})+\sigma^{-2}(z_i\text{-}a\text{-}b\eta_i^{(c)})^2+\eta_i^{(c)\prime}\eta_i^{(c)}\right\}\right]}\right\} \qquad (2.25)$$

If $\alpha \geq u$ where u has a random uniform distribution, the transition jump $\eta_i^{(c)} \rightarrow \eta_i^{(c+1)}$ is allowed, otherwise the jump does not occur and the current value is retained in the Markov chain position.

We chose for our proposal density a normal distribution centered on the current value $\eta_i^{(c)}$. The scale and spread of the proposal density are important factors controlling the acceptance or rejection rate and the sample space region covered by the chain. For accuracy it is desirable that the density be sampled mostly around its mode. If the variance of the density is too large some generated candidates will be too far from the mode and so have relatively low acceptance probability. On the other hand if the variance is too small, it will prolong the time required by the process to sufficiently traverse the sampling space supported by the density, leading to under-sampling of the low probability regions. To achieve a delicate balance an approximate acceptance rate of 0.45 is recommended when dealing with one-dimensional problem like ours where we estimate only one 'parameter' (η_i) (Chib & Greenberg, 1995). Therefore a proper fine tuning of the variance of proposal density is necessary for good mixing and efficient sampling (Chib & Greenberg, 1995; Walsh, 2004). As a rough guide, we compute the Empirical Bayes' variance estimate of $[\eta \mid \boldsymbol{y};\boldsymbol{\Omega}^{(k)}]$ for use as a start value. From general multivariate results for factor analytic model, the latent factors conditional on the observed indicators are multivariate normal: $[\eta \mid \mathbf{y},\boldsymbol{\Omega}] \sim \mathrm{N_m}(\boldsymbol{\mu}_{\eta|\mathbf{y}},\boldsymbol{\Sigma}_{\eta|\mathbf{y}})$. Therefore given a standard normal marginal density of η (see (2.7) to (2.9)), the common conditional variance is computed as follows

$$\boldsymbol{\Sigma}_{\eta|y} = \boldsymbol{I} - \boldsymbol{\Lambda}'(\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \boldsymbol{\Theta})^{-1}\boldsymbol{\Lambda} \tag{2.26}$$

2.3.2 The Maximization Step of EM

Recall the decomposition of the complete data log-likelihood (see (2.12) and (2.13)) as reproduced below:

$$\begin{aligned} Q(\boldsymbol{\Omega}, \boldsymbol{\Omega}^{(k)}) &= E^{\eta}\left[[\log p(\mathbf{y} \mid \eta; \boldsymbol{\Omega}_y) + \log p(z|\eta; \boldsymbol{\Omega}_z) + \log p(\eta)] \mid \mathbf{y}, z, \boldsymbol{\Omega}^{(k)}\right] \\ &\propto E^{\eta}[\log p(\mathbf{y} \mid \eta; \boldsymbol{\Omega}_y) \mid \mathbf{y}, z, \boldsymbol{\Omega}^{(k)}] + E^{\eta}[\log p(z|\eta; \boldsymbol{\Omega}_z) \mid \mathbf{y}, z, \boldsymbol{\Omega}^{(k)}] + \log p(\eta) \\ &\propto \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}[\log p(\mathbf{y_i} \mid \eta_i^{(c)}; \boldsymbol{\Omega}_y) \mid \mathbf{y}, \boldsymbol{\Omega}^{(k)}] \quad + \quad \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}[\log p(z_i|\eta_i^{(c)}; \boldsymbol{\Omega}_z) \mid z, \boldsymbol{\Omega}^{(k)}] \quad + \quad w \end{aligned} \tag{2.27}$$

The first two terms on the right of (2.27) on the first line (a factor analytic model and a univariate regression model) have their separate distinct parameters, so maximization can be done separately (McLachlan & Krishnan, 2008). Note that for the purpose of identification the marginal distribution $p(\eta)$ has 0 mean and unit variance (Dempster, Laird, & Rubin, 1977). Secondly, even if we resort to approximating $p(\eta)$ by its conditional distribution in the M-step, this will not be useful since $[\eta \mid \boldsymbol{y}, z; \boldsymbol{\Omega}^{(k)}]$ can be specified only up to a normalizing constant (see (2.22)) and the conditional distribution is proportional to its joint distribution. Therefore the last term is treated here as a constant *w* (2.27) that does not depend on Ω hence does not contribute to the maximization. The EM algorithm hence concerns the finding of

1. $\boldsymbol{\Omega}_y^{(k+1)}$ to maximize $E^{\eta}[\log p(\boldsymbol{y} \mid \eta; \boldsymbol{\Omega}_y) \mid \boldsymbol{y}, \boldsymbol{\Omega}_y^{(k)}]$, and
2. $\boldsymbol{\Omega}_z^{(k+1)}$ to maximize $E^{\eta}[\log p(z|\eta; \boldsymbol{\Omega}_z) \mid z, \boldsymbol{\Omega}_z^{(k)}]$.

An alternative approach to maximization is based on the idea of a Stochastic EM algorithm as described in (Lee, Song, & Lee, 2003). Here the mean of random observations ($\hat{\eta}_i$) in the

Markov chain for the *i*th subject is computed, considered as fixed, and simultaneous regression model is solved to obtain mle's. For example, their approach to modeling the Y's would give:

$$\hat{\boldsymbol{\Omega}}_{\mathbf{y}}^{(\mathbf{k}+1)} = \arg\max_{\boldsymbol{\Omega}_{\mathbf{y}}} Q(\boldsymbol{\Omega}_{\mathbf{y}}, \boldsymbol{\Omega}^{(\mathbf{k})}) = \arg\max_{\boldsymbol{\Omega}_{\mathbf{y}}} [\log p(\mathbf{y} \mid \hat{\eta}; \boldsymbol{\Omega}_{\mathbf{y}}) \mid \mathbf{y}, \boldsymbol{\Omega}_{\mathbf{y}}^{(\mathbf{k})}],$$

$$\hat{\eta}_i = \frac{1}{C} \sum_{c=1}^{C} \eta_i^{(c)}, \quad i=1,.....N$$

However our approach to maximization is slightly different in the sense that we plugged $\eta_i^{(c)}$ directly into the regression model and solve *C* simultaneous regression equations instead. A major consideration in our decision is to avoid bias in our estimation, since the computed likelihoods from the two methods are not necessarily equivalent. Our approach requires the assumption that the random sample elements in each Markov chain (subject) are independent; which we are able to satisfy by using thinning method as necessary to minimize autocorrelation. Secondly we can also assume independent observations between subjects since the C Markov samples are independently generated for each subject to yield N independent Markov chains. So, using the MCMC method, the kth M-step solves

$$\begin{aligned} \hat{\boldsymbol{\Omega}}_{\mathbf{y}}^{(\mathbf{k}+1)} &= \arg\max_{\boldsymbol{\Omega}_{\mathbf{y}}} Q(\boldsymbol{\Omega}_{\mathbf{y}}, \boldsymbol{\Omega}^{(\mathbf{k})}) = \frac{1}{C} \sum_{c=1}^{C} \arg\max_{\boldsymbol{\Omega}_{\mathbf{y}}} [\log p(\mathbf{y} \mid \eta^{(c)}; \boldsymbol{\Omega}_{\mathbf{y}}) \mid \mathbf{y}, \boldsymbol{\Omega}_{\mathbf{y}}^{(\mathbf{k})}] \\ \hat{\boldsymbol{\Omega}}_{\mathbf{z}}^{(\mathbf{k}+1)} &= \arg\max_{\boldsymbol{\Omega}_{\mathbf{z}}} Q(\boldsymbol{\Omega}_{\mathbf{z}}, \boldsymbol{\Omega}^{(\mathbf{k})}) = \frac{1}{C} \sum_{c=1}^{C} \arg\max_{\boldsymbol{\Omega}_{\mathbf{z}}} [\log p(z \mid \eta^{(c)}; \boldsymbol{\Omega}_{\mathbf{z}}) \mid z, \boldsymbol{\Omega}_{\mathbf{z}}^{(\mathbf{k})}] \end{aligned} \tag{2.28}$$

One notable advantage of the ALV model structure is that with η_i available, the two parts above have fixed-effects GLM structure and maximum likelihood estimation can be carried out separately for them using the existing statistical tools for standard regression models. For the future ALV model extensions, all that is required of either part is for the response variables to belong to the exponential family.

2.3.3 Standard Errors of Estimates

In the context of EM algorithm the standard errors of the maximum likelihood estimates $\hat{\boldsymbol{\Omega}}$ may be obtained from the inverted Hessian or information matrix based on the observed data likelihood function, according to the method of Louis e.g. (Lee & Zhu, 2002; Song & Lee, 2005; Law, Taylor, & Sandler, 2002). The Louis method expresses observed information matrix as the difference between complete and missing data information matrices, thus

$$\begin{aligned} \mathrm{I}(\hat{\boldsymbol{\Omega}};\mathbf{y},\mathrm{z}) = \quad & \mathrm{E}^{\eta}\left(-\frac{\partial^2}{\partial\boldsymbol{\Omega}\,\partial\boldsymbol{\Omega}'}\log \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega};\mathbf{y},\mathrm{z},\eta)\,|\,\mathbf{y},\mathrm{z},\hat{\boldsymbol{\Omega}}\right) \\ - \quad & \mathrm{Var}\left(\frac{\partial}{\partial\boldsymbol{\Omega}}\log \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega};\mathbf{y},\mathrm{z},\eta)\,|\,\mathbf{y},\mathrm{z},\hat{\boldsymbol{\Omega}}\right) \end{aligned} \tag{2.29}$$

where $\mathrm{I}(\hat{\boldsymbol{\Omega}};\boldsymbol{y},\mathrm{z})$ is the observed information and the first and second terms on the right represent complete and missing data information evaluated at the final parameter maximum likelihood estimates $\hat{\boldsymbol{\Omega}}$. This approach is chosen because the EM implementation does not generate observed data information as a by-product. However since these matrices generally have no closed forms, the Louis' method provides a formula for computing the observed information matrix in terms of the expectation of the first and second derivatives of the complete data log likelihood function using the MCMC samples (McLachlan & Krishnan, 2008). The missing data information formula is written as

$$\begin{aligned} \mathrm{I}(\hat{\boldsymbol{\Omega}};\mathbf{y},\mathrm{z}) \approx \quad & \frac{1}{\mathrm{C}}\sum_{\mathbf{c}=1}^{\mathrm{C}}\frac{-\partial^2\log \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega};\mathbf{y},\mathrm{z},\eta^{(\mathrm{c})}\,|\,\mathbf{y},\mathrm{z},\hat{\boldsymbol{\Omega}})}{\partial\boldsymbol{\Omega}\,\partial\boldsymbol{\Omega}'} \\ + \quad & \frac{1}{\mathrm{C}}\sum_{\mathbf{c}=1}^{\mathrm{C}}\left\{\frac{\partial\log \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega};\mathbf{y},\mathrm{z},\eta^{(\mathrm{c})}\,|\,\mathbf{y},\mathrm{z},\hat{\boldsymbol{\Omega}})}{\partial\boldsymbol{\Omega}} - \frac{1}{\mathrm{C}}\sum_{\mathbf{c}=1}^{\mathrm{C}}\frac{\partial\log \mathrm{L}_{\mathrm{com}}(\boldsymbol{\Omega};\mathbf{y},\mathrm{z},\eta^{(\mathrm{c})})\,|\,\mathbf{y},\mathrm{z},\hat{\boldsymbol{\Omega}}}{\partial\boldsymbol{\Omega}}\right\}^2 \end{aligned} \tag{2.30}$$

Given the availability of η and assuming normally distributed response variables, the complete-data log likelihood function and related partial derivatives can be easily obtained separately for each outcome variable at each point in the Markov chain in the form of a least square regression model:

$$\begin{aligned}
\log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}}) &= -\frac{1}{2}N\log(2\pi) - \frac{1}{2}N\log\hat{\boldsymbol{\theta}} - \frac{1}{2}\hat{\boldsymbol{\theta}}^{-1}\sum_{\mathbf{i}=1}^{\mathbf{N}}(\mathbf{y_i} - \hat{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\lambda}}\eta_i^{(c)})^2 \\
\frac{\partial \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \hat{\boldsymbol{\upsilon}}} &= \hat{\boldsymbol{\theta}}^{-1}\sum_{\mathbf{i}=1}^{\mathbf{N}}(\mathbf{y_i} - \hat{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\lambda}}\eta_i^{(c)}) \\
\frac{\partial \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \hat{\boldsymbol{\lambda}}} &= \hat{\boldsymbol{\theta}}^{-1}\sum_{\mathbf{i}=1}^{\mathbf{N}}\eta_i(\mathbf{y_i} - \hat{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\lambda}}\eta_i^{(c)}) \\
\frac{\partial \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \hat{\boldsymbol{\theta}}} &= -\frac{1}{2}N\hat{\boldsymbol{\theta}}^{-1} + \frac{1}{2}N\hat{\boldsymbol{\theta}}^{-2}\sum_{\mathbf{i}=1}^{\mathbf{N}}\eta_i(\mathbf{y_i} - \hat{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\lambda}}\eta_i^{(c)})^2
\end{aligned} \tag{2.31}$$

where $\hat{\boldsymbol{\Omega}}_{\mathbf{y}} = \{\hat{\upsilon}_1,...,\hat{\upsilon}_p,\hat{\lambda}_1,...,\hat{\lambda}_p,\hat{\theta}_1,...,\hat{\theta}_p\}$ is the set of MLE's associated with *p* indicator variables. The corresponding second partial derivatives are

$$\begin{aligned}
\frac{\partial^2 \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \boldsymbol{\upsilon}^2} &= -\hat{\boldsymbol{\theta}}^{-1} \\
\frac{\partial^2 \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \boldsymbol{\lambda}^2} &= -\hat{\boldsymbol{\theta}}^{-1}\sum_{\mathbf{i}=1}^{\mathbf{N}}(\eta_i^{(c)})^2 \\
\frac{\partial^2 \log L_{com}(\hat{\boldsymbol{\Omega}}_{\mathbf{y}})}{\partial \boldsymbol{\theta}^2} &= \frac{1}{2}N\hat{\boldsymbol{\theta}}^{-2} - N\hat{\boldsymbol{\theta}}^{-3}\sum_{\mathbf{i}=1}^{\mathbf{N}}\eta_i^{(c)}(\mathbf{y_i} - \hat{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\lambda}}\eta_i^{(c)})^2
\end{aligned} \tag{2.32}$$

Appropriate combinations of the above derivations according to the Louis' formula will supply the approximate ingredients of the observed information matrix with respect to each outcome. Similar expression can be derived for the Z variable.

According to the literature, out of the available different techniques for computation of the standard errors within the EM setting, Louis' method is best suited for adaptation to the

Monte Carlo version of the EM (McLachlan & Krishnan, 2008), our simulation results show values that are much smaller than the true population values. Therefore we have decided to explore a different option that is more applicable to our situation. According to (Gamerman & Lopes, 2006), if the ergodic theorem is applied where it is possible to go from every state to every other state, the summaries (marginal point or intervals) of any real function $\boldsymbol{\Omega} = t(\eta)$ can be consistently estimated by their corresponding estimators based on the generated random sample. If for each state of the Markov chain we have $\boldsymbol{\Omega}^c = t(\eta^c)$, then we can estimate the posterior mean and variance of $\boldsymbol{\Omega}$ by

$$\begin{aligned} \hat{E}(\boldsymbol{\Omega}) &= \hat{\boldsymbol{\Omega}} = \frac{1}{C}\sum_{c=1}^{C}\boldsymbol{\Omega}_c = \frac{1}{C}\sum_{c=1}^{C} t(\eta_c) \\ \hat{Var}(\boldsymbol{\Omega}) &= \frac{1}{C}\sum_{c=1}^{C}(\boldsymbol{\Omega}_c - \hat{\boldsymbol{\Omega}})^2 \end{aligned} \tag{2.33}$$

Since we are employing standard regression model estimation techniques, the ML estimates as well as their standard errors are direct products of regression analyses and can be treated as functions of η. Furthermore, in addition to the theoretical support for the use of the ergodic averages as estimates, the approximate confidence intervals about these estimates can be computed based on central limit theorem (Gamerman & Lopes, 2006). For example, for C = 1000, the 0.025 and 0.975 quantiles of $\boldsymbol{\Omega}$ can be consistently estimated by the limits provided by the 25^{th} and 975^{th} largest sample values of $\boldsymbol{\Omega}$. In summary, based on the results of our simulation studies, the standard error estimates from this approach are relatively closer to the true values compared to the estimates obtained using the Louis' method.

CHAPTER 3

ADDITIVE MODEL COMPONENT OF ALV MODEL

3.1 General Introduction

In this chapter and the next we formally introduce the theoretical basis for the integration of Generalized Additive Models (GAM) into a latent variable framework to model a smoothed nonlinear relationship between the latent baseline risk and a distal outcome. Specifically, this requires the development of a statistical model linking the latent variable η to the outcome Z with a spline function. So we start the current chapter by first exploring further the crucial role played by Additive Models (AM) in the assessment of variation in intervention impact. We then devote the rest of the chapter to discussing the basic theory and estimation of GAM along that line.

3.2 The Additive Model

Consider a set of independent observations $\{X_1,....X_p, Z\}$, consisting of response random variable *Z* and a set of predictor variables X which may include interaction terms. The standard linear regression model fit to the data is specified as

$$\mu = \alpha + \sum_{j=1}^{p} x_j \beta_j \tag{3.1}$$

where $\mu = \mathrm{E}(\mathrm{Z})$ and β_j is unknown parameter or coefficient that quantifies the dependence of the μ on x_j. Let us express the linear regression problem as $\mathrm{Z_i} = \mathrm{s}(\mathrm{x_i}) + \mathrm{e_i}, \mathrm{i} = 1,...,\mathrm{n};$

$x_i \in [0,1]; e_i \sim N(0,\sigma^2)$. Then $s(.)$ is assumed to be of the form $s(x,\beta)$, linear in β and known up to the parameters β to be estimated from the data. Generally in parametric models including both linear and nonlinear regression models, $s(.)$ is constrained such that the model space dimension (number of unknown parameters in β) is much smaller than *n*, with consequent possible model misspecification. In contrast, non-parametric and semi-parametric estimation methods allow $s(.)$ varying in a relatively high dimension function state to avoid possible model misspecification (Gu, 2002). One such popular method is the Additive Model (AM) (Hastie & Tibshirani, 1990) which is a generalization of the linear regression model that allows for a more flexible description of the dependence of the outcome *Z* on individual predictor term in *X,* without requiring the usual rigid parametric form for the dependence. Although a standard multiple regression model is additive in nature but there is a single coefficient β per predictor term to explain its relationship, which is very restrictive. The idea in AM is that a complex nonlinear relationship often requires the estimation of more than a single coefficient for each predictor in order to achieve the best prediction of the outcome variable values.

The AM accomplishes this by estimating an unspecified or non-parametric (smoothing) function for each covariate to produce a representation of the trend of *Z* (as a function of one or more predictors) that is less variable than *Z* itself. Here the estimated smooth functions serve as analogues of the coefficients in standard linear model (Hastie & Tibshirani, 1990):

$$\mu = \alpha + \sum_{j=1}^{p} s_j(x_j) \tag{3.2}$$

The functions are fitted using scatterplot smoothers which are nonparametric techniques that define the relationships between the response variable and each predictor in a flexible way,

thereby relieving the user from the need to search for the appropriate transformation for each predictor (Chambers & Hastie, 1993). Note that the two models described above are both additive in their predictor effects which also can be examined separately, in the absence of interactions. However in AM the linear predictors are replaced with additive predictors which are represented as smooth functions of the predictors (Hastie & Tibshirani, 1990). There are multivariate assumptions underlying the AM and hence the name *additive*: a low-dimensional additive structure to the p-predictor function of *X*, that are "far easier to interpret than a p-dimensional multivariate surface" (Chambers & Hastie, 1993). So an additive approximation to the multivariate regression function is obtained by using a univariate smoother to estimate the individual additive terms; this way the problem of 'curse of dimensionality' (e.g. rapidly increasing variance with increasing dimensionality) is avoided (Xiang, 2004).

Similarly, the Generalized Additive Model (GAM) (Hastie & Tibshirani, 1990; Wood, 2006) and the Generalized Additive Mixed Model (Wood, 2006) are the respective 'additive' or smooth nonlinear versions of the Generalized Linear Model (GLM) and the Generalized Linear Mixed Model (GLMM). Also, a semi parametric form of the AM can be fitted whereby a linear relation of the outcome with some predictors is assumed while unknown functional relations are assumed for some other predictors in the model and are explored by smoothers. For example, a GAM that has *p* non-parametric smooth functions of η plus *k* parametric terms can be expressed in the general form

$$g(\mu) = \alpha + \sum_{j=1}^{p} s_j(\eta_j) + \sum_{m=1}^{k} x_m \beta_m \qquad (3.3)$$

where $\mu_i = E(Y_i)$ and $Y_i \sim$ some distribution in the exponential family.

From another perspective (Holler, 2005), GLMs may be seen as a special case of GAMs. For example consider a regression equation of the form $\mu = Intercept + s_1(x1) + s_2(x2)$ as a generic additive model. For a GLM the functions s_1 and s_2 can be polynomial, categorical, or transforms e.g. log. In a GAM one or both functions may be represented as non-parametric smoothers; in the former case we have the semi parametric type of GAM. The question then is how to strike the best balance between the degrees of freedom, amount of data, and functional form (Holler, 2005).

3.3 Baseline-Treatment Interactions using GAM

As already indicated in the first chapter, additive models (GAM, GAMM) are particularly useful for uncovering a potential nonlinear structure between an outcome and each continuous covariate (and its interaction with other predictors) that one might otherwise miss. Consider a GAM modeling of the dependence of the mean of the outcome Z on treatment Tx (intervention=1, control=0), and the smooth functions of the baseline risk covariate η and baseline-treatment interaction:

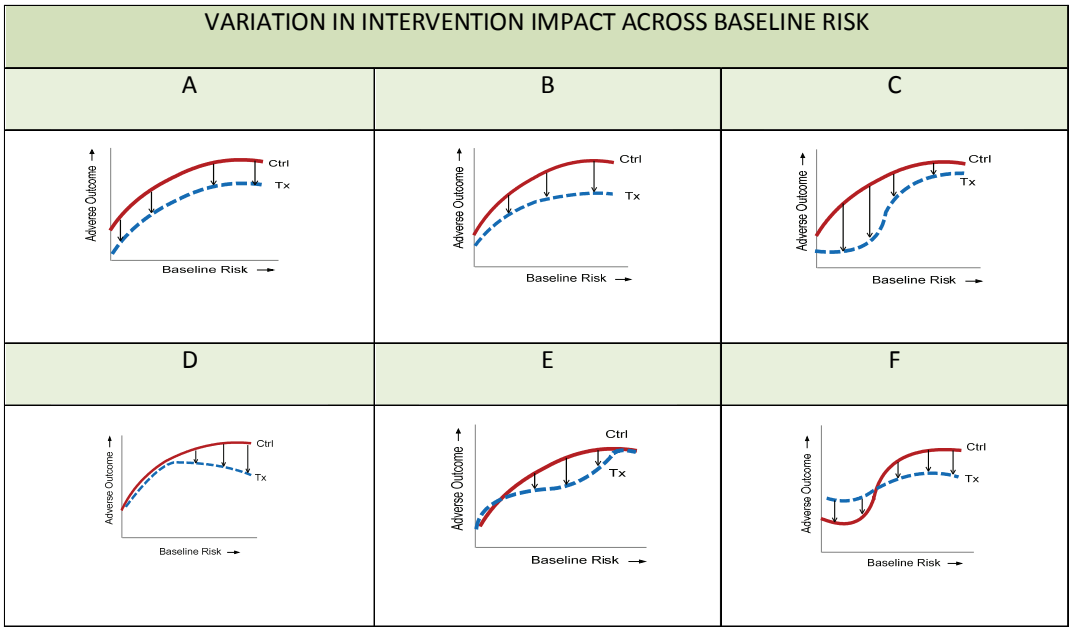
$$g(E[z_i]) = \alpha + s_1(\eta_i) + \beta(Tx_i) + s_2(\eta_i * Tx_i) \quad (3.4)$$

In addition to the use of smoothers by the GAM procedure to estimate the dependence in the data based on the model, the smoothers are also used to estimate the distribution shapes to enhance the visual appearance of the plot of Z against the predictor (Hastie & Tibshirani, 1990), and to describe vividly the nature of the treatment-baseline interaction (Brown, 1993; Khoo, 1997; Brown, et al., 2008). The usefulness of these models can be best illustrated with hypothetical situations such as described in the plots in figure 3.1 which is modified from Khoo (1997) with additions. In the plots, Y is the fitted outcome of a GAM model in which Z is regressed on the

treatment variable plus smooth functions of baseline risk and baseline-treatment interaction. On the x-axis the level of baseline risk increases from left to right. The dashed curve represents the treatment that is designed to reduce outcome Z relative to the control (solid line). Any tangible separation between the two curves indicates intervention effects. The length of a vertical arrow measures the drop in Z along y-axis, thereby depicting the magnitude of intervention effects at a given level of baseline risk. Generally all the plots display a nonlinear increase of the outcome with the baseline risk irrespective of the intervention condition.

In plot A the curves are parallel and the constant length of the arrows indicates constant treatment effects across all levels of the baseline risk; hence there is no baseline-treatment interaction. In contrast there is a steady or linear increase of group difference (drop arrows) in Z as the baseline risk increases in plot B; this signifies a linear baseline-treatment interaction. The higher the baseline risk levels of the subject the more effective the intervention. In plot C the Z drop arrow length initially increases with baseline then tapers off; that is, the intervention is effective for individuals in the lower end of the risk scale but less so for the high risk individuals. The opposite occurs in plot D where the intervention is rather effective for only the high risk individuals. Plot E describes a rather interesting situation where the intervention impact is most effective in some middle region but not at the extremes of risk. Such situations exist whenever too little or too much of a baseline characteristic that is the target of intervention is problematic and more resistant to modification. For example, either extreme on a parenting scale (too authoritative or too permissive) may lead to poorer child outcomes than moderate scores on this scale. Lastly, it is not uncommon that program interventions may produce iatrogenic effects. As shown in plot F, the intervention is detrimental to low risk individuals but the impact gradually shifts to being beneficial as the baseline risk level gets higher.

Figure 3.1 The Plots of Distal Outcome versus Baseline Risk



As we can see, analyses that ignore variation in intervention impact may not be telling the whole story since all the hypothetical situations depicted on the plots are not implausible. Apart from gaining insight as to what works and for whom, we may also uncover unintended consequences of a given intervention if there is any. Much of this obtainable extra information is contingent on the ability to capture the nonlinear outcome-baseline relationship; this type of information may be easily lost if we are limited to linear modeling techniques. Most importantly, while GAM type models are most suitable for exposing such nonlinear dependence in the data, they can handle linearity as well.

3.4 The Best Smoothing Function

Motivation

For a simple illustration consider this time a set of independent bivariate observations consisting of outcome *Z* and predictor η, where $\{\eta_i, z_i\}$, $i = 0, 1, 2, \ldots, n$ and $a = \eta_0 < \eta_1 < \eta_2 < \ldots \eta_n = b$. We wish to fit a curve through the data points and plot it on a graph, infer data values between the data points and estimate some parameters from the data. Suppose we wish to fit a simple function $s(\eta)$ that can be easily manipulated to the discrete data, such that it matches the data points exactly; such a function will be an interpolant. Some families of common interpolant functions include polynomials, piecewise polynomials or splines (segments of polynomials joined together at data points or knots); trigonometric, exponential and rational functions.

Polynomials are popular candidates for interpolation because they are continuously differentiable up to all orders so that the smoothness can be easily quantified. However, simply fitting a single high-degree polynomial function to several data points is plagued with excessive oscillations thereby resulting in some misfit. For this reason polynomial bases are not efficient for representing $s(\eta)$ when we are interested in the whole domain of $s(\eta)$ (Wood, 2006). A better alternative is to employ a piecewise polynomial interpolation (spline bases) which allows for fitting low-degree polynomials (e.g. cubics) to interval segments on the η continuum (Heath, 2005; Wood, 2006; Cheney & Kincaid, 2004). So in terms of fitting a model to sampled data from a function, the idea is to create a spline that approximates that data well. Therefore it is necessary to determine which of the low order polynomials will be most appropriate for achieving optimal smoothness and minimal error. While choosing the best curve fitting function is of paramount importance it should not be done arbitrarily (Wood, 2006).

Smoothness Property

If we assume that the data points represent a discrete sample of an underlying continuous function, fitting all the observed points exactly may be undesirable because the behavior of the function spanning the discrete data points will likely be highly variable. For example the results of plotting a candidate function may be unpleasing to the eyes because of excessive oscillations or sharp curvatures. A curvature is a function of the second derivative (rate of change of slope) at the given data point. Therefore for $s(\eta)$ to be the best smoothing interpolant, it must possess the minimum magnitude of the integrated squared second derivatives over all data points, that is, $\min \int_a^b [s''(\eta)]^2 \partial\eta$ from among all other interpolating functions (that are differentiable up to second derivatives) over the same set of data points. Let

$$z_i = s(\eta_i) + e_i, \quad E(e_i) = 0, \quad Var(e_i) = \sigma^2 < \infty,$$

We wish to estimate the unknown function $s(\eta)$ without specifying a form for s except that s belongs to a class of suitably smooth functions. So in terms of data fitting we seek a general solution to the penalized least squares criterion (least squares criterion with respect to 'optimal smoothness'), specified as

$$\sum_i [z_i - s(\eta_i)]^2 + \tau \int_a^b [s''(\eta)]^2 \partial\eta \tag{3.5}$$

where τ is the smoothing parameter. The first term in (3.5) measures approximation to the data, and the second term controls smoothness by penalizing larger curvatures.

Theorem (Cheney & Kincaid, 2004; Heath, 2005)

For a given τ, there exists an interpolant $s(\eta)$ for the set $\{\eta_i, z_i\}$, where, of all twice-continuously differentiable functions $f(\eta)$ that interpolate $\{\eta_i, z_i\}$, $s(\eta) \in f(\eta)$ is the smoothest interpolant, i.e. an explicit, unique minimizer of (3.5) in the sense of having the smallest integrated squared second derivative over $\{\eta_i, z_i\}$. Thus we have the following *Lemma*:

$$\int_a^b s''(\eta)^2 \partial\eta \le \int_a^b f''(\eta)^2 \partial\eta \tag{3.6}$$

Proof

We need to show that if certain conditions are satisfied, $s(\eta)$ will qualify as the smoothest interpolant. Since $s(\eta)$ and any other $f(\eta)$ are interpolants with knots at all the data points in the interval, the functions must be equal at η_i; hence it follows that $f(\eta_i) = s(\eta_i)$ and also

$$\sum_i [z_i - f(\eta_i)]^2 = \sum_i [z_i - s(\eta_i)]^2 .$$

Therefore we let $g(\eta) = f(\eta) - s(\eta) = 0$ such that $f'' = s'' + g''$. By expansion

$$\int_a^b (f'')^2 \partial\eta = \int_a^b (s'')^2 \partial\eta + 2\int_a^b s''g'' \partial\eta + \int_a^b (g'')^2 \partial\eta .$$

Note that we are mainly interested in the magnitudes of the integrated squared second derivatives. Hence we see that the inequality $\int_a^b [s''(\eta)]^2 \partial\eta \le \int_a^b [f''(\eta)]^2 \partial\eta$ will be true if the integral $\int_a^b s''g'' \partial\eta = 0$, so that

$$\int_a^b (f'')^2 \partial\eta = \int_a^b (s'')^2 \partial\eta + \int_a^b (g'')^2 \partial\eta \geq \int_a^b (s'')^2 \partial\eta$$

Therefore, to prove our theorem we next need to show that this integral equals zero under certain specified conditions which must be satisfied by $s(\eta)$. We set out to accomplish the task by integrating by parts. Using the formula $\int u\partial v = uv - \int v\partial u$, let $s'' = u, \quad g''\partial\eta = \partial v$, then we have

$$\int_a^b s''g''\partial\eta = s''g'|_a^b - \int_a^b s'''g'\partial\eta .$$

Now, the first set of conditions is: $s''(a) = 0$ and $s''(b) = 0$, that is, the second derivatives at both end points $a = x_1$ and $x_n = b$ must be set to zero. This done, we will then have

$$\int_a^b s''g''\partial\eta = -\int_a^b s'''g'\partial\eta$$

If we break the interval $[a, b]$ into its n-1 segments of component functions joined together at the knots, the equation becomes discrete summation over all segments, that is

$$\int_a^b s''g''\partial\eta = -\int_a^b s'''g'\partial\eta = \sum_{i=1}^{n-1} \int_{\eta_i}^{\eta_{i+1}} s'''g'\partial\eta$$

Next, it is required that s''', the 3rd derivative at each unit interval $[\eta_i, \eta_{i+1}]$ be a constant, say c_i, a property of cubic polynomial at each interval, so that we have

$$\sum_{i=1}^{k} \int_{x_i}^{x_{i+1}} S'''g'\partial x = \sum_{i=1}^{n-1} \int_{\eta_i}^{\eta_{i+1}} c_i g'\partial\eta = \sum_{i=1}^{n-1} c_i \int_{\eta_i}^{\eta_{i+1}} g'\partial\eta = \sum_{i=1}^{n-1} c_i \left[g(\eta_{i+1}) - g(\eta_i) \right] = 0 .$$

The last term above equals zero because we specify at the beginning that $g(\eta_i) = 0$ for every knot, thus the proof.

So far we have determined a number of conditions that must be imposed on $s(\eta)$ for it to be the smoothest interpolant: be a cubic spline with knots at the unique values of η, and the second derivatives at the end points set to zero. By definition, the function that satisfies these conditions is a natural cubic spline (Cheney & Kincaid, 2004).

3.5 Cubic Splines

There are several types of splines in the literature and the typology may be associated with how the splines are represented, the spacing of the knots, and type of other conditions imposed. For example, in B-splines basis functions are used for the entire spline, interpolating splines require that the splines include some given values, zero second derivatives are enforced at the end knots to yield natural splines, and uniform splines have evenly spaced knots; just to mention a few. As already shown, the natural cubic splines are the best available curve fitting functions (Cheney & Kincaid, 2004; Wood, 2006).

A *k*-degree spline function is a function consisting of *k*-degree polynomial pieces joined together and are continuously differentiable *k*-1 times (Heath, 2005). A cubic spline (*k* = 3) is a twice continuously differentiable piecewise polynomial function. The connection points of the polynomial pieces plus the two end points are known as the knots of the spline. The polynomials join smoothly at these knots because the cubic spline is continuous up to second derivative across the knots (Wood, 2006).

Supposing an N-vector η (single predictor variable) is divided into *n* intervals so that $\eta_0 < \eta_1 < \eta_2 < \ldots < \eta_n$ represent $n+1$ unique values. Let different cubic polynomials be fitted to each interval $\left[\eta_j, \eta_{j+1}\right]; \quad j = 1, \ldots, n$. In its standard representation the knots of a cubic spline

coincide exactly with the unique values of η in the data; and the 1st and 2nd derivatives including the values of the cubic spline at the knots are specified to yield a number of equations and polynomial coefficients (parameters) to be estimated. Each cubic polynomial piece joins two adjacent knots and has four unknown coefficients β's whose values vary from one piece to the other. Given *n* intervals in the piecewise polynomial, there are *n*+1 (or *q*) knots, thus there are *n* different cubics and 4*n* spline coefficients in all. The estimates of the coefficients are therefore simultaneous solutions to a system of linear equations. To get a unique set of solution, it is required that the number of equations and parameters be equal. Thus for a simple standard representation of a natural cubic spline to be fitted to the set of *n*+1 knots, a total number of 4*n* equations is formed with continuity conditions imposed on the cubic polynomials as listed in Table 3.1 below (Heath, 2005; Cheney & Kincaid, 2004):

Table 3.1 Cubic Spline Interpolation

	Three Continuity Conditions	**Number of equations**
1	Each cubic to pass through the 2 knots at either end of its interval $(\eta_j,\ \eta_{j+1})$	2n
2	1st derivatives of adjacent cubics to match at each of n-1 interior knots $\eta_j\ (j \neq 0,n)$	n-1
3	2nd derivatives of adjacent cubics to match at each of n-1 interior knots $\eta_j\ (j \neq 0,n)$	n-1
4*	2nd derivatives of first and last cubics to be fixed at zero at endpoints η_0 and η_n	2
	Total number of equations	4n

* addition of this specification results in a natural cubic spline function

Following an example that is illustrated in (Heath, 2005), suppose we wish to estimate the natural cubic spline function that interpolates three data points $\{\eta_j,\ z_j\}$, $j = 0, 1, 2$. So we have $n = 2$

intervals $(\eta_0, \eta_1), (\eta_1, \eta_2)$ with two cubic polynomials joined at the 3 knots to represent the cubic spline; and $4n = 8$ simultaneous equations to estimate 8 parameters a, b in the two polynomials denoted as

$$\begin{aligned} p_1(\eta) &= a_1 + a_2\eta + a_3\eta^2 + a_4\eta^3 \\ p_2(\eta) &= b_1 + b_2\eta + b_3\eta^2 + b_4\eta^3 \end{aligned} \tag{3.7}$$

The $2n = 4$ equations satisfying continuity condition 1 in the table 3.1 are specified as follows:

$$\begin{aligned} \text{At } \eta_0 : &\quad a_1 + a_2\eta_0 + a_3\eta_0^2 + a_4\eta_0^3 = z_0 \\ \text{At } \eta_1 : &\quad a_1 + a_2\eta_1 + a_3\eta_1^2 + a_4\eta_1^3 = z_1 \\ \text{At } \eta_1 : &\quad b_1 + b_2\eta_1 + b_3\eta_1^2 + b_4\eta_1^3 = z_1 \\ \text{At } \eta_2 : &\quad b_1 + b_2\eta_2 + b_3\eta_2^2 + b_4\eta_2^3 = z_2 \end{aligned} \tag{3.8}$$

Condition #2 requires the first derivatives of the two polynomials to match at the lone interior point:

$$\text{At } \eta_1 : \quad a_2 + 2a_3\eta_1 + 3a_4\eta_1^2 = b_2 + 2b_3\eta_1 + 3b_4\eta_1^2 \tag{3.9}$$

Similarly, condition #3 with respect to the second derivatives gives the equation:

$$\text{At } \eta_1 : \quad 2a_3 + 6a_4\eta_1 = 2b_3 + 6b_4\eta_1 \tag{3.10}$$

The final two equations satisfying the 4th condition relate to the endpoints:

$$\begin{aligned} \text{At } \eta_0 : &\quad 2a_3 + 6a_4\eta_0 = 0 \\ \text{At } \eta_2 : &\quad 2b_3 + 6b_4\eta_2 = 0 \end{aligned} \tag{3.11}$$

The above representations and notations are for the very basic conventional spline where the knots coincide exactly with the input data points. Typically less number of knots than data

points are chosen and may be evenly spaced over the range of values of η that is constrained to between 1 and 0 (Wood, 2006). Alternatively the knots may be placed at the quintiles of unique values distribution of η. We will revisit how to determine the optimal number of knots later in this chapter.

3.5.1 Representation of Natural Cubic Splines

A critical objective of GAM fitting is ensuring that the chosen smoothing function is the best smoother, as well as fits or summarizes the data well. This property is related to how the smooth function is represented. The representation of the smoothing function can take many forms and can be very complicated and intimidating especially for those forms that are most suitable for computation and general practical use. Therefore, representing the smooth functions and choosing how smooth the functions should be are two critical issues of major theoretical importance in additive modeling (Hastie & Tibshirani, 1990; Wood, 2006). There is more than one approach to representing GAM depending on the method of estimation. The estimation by backfitting technique (Hastie & Tibshirani, 1990) iteratively fits each smoothing component to its partial residuals until the individual components no longer change (convergence) but automatic smoothness selection is very costly (Wood, 2006). Another approach to estimation is penalized regression splines; this involves choosing some basis functions defined as the space of functions of which the smoothing function is an element (Wood, 2006). Here the degree of smoothness of model terms is estimated as part of the GAM algorithm (Wood, 2006), therefore we prefer this latter approach for our work. The estimation of degree of smoothness is not integrated into the backfitting procedure (Wood, 2006) and the degree has to be chosen by the user.

To illustrate the basic principles, we will again use a simple regression model of the outcome *Z* with a smooth function of the single predictor η :

$$z_i = s(\eta_i) + \varepsilon_i \tag{3.12}$$

A proper representation of (3.12) requires that it becomes a linear model. One way to achieve this is by choosing for $s(.)$ some basis functions and treating them as known (Wood, 2006):

$$s(\eta) = \sum_{l=1}^{L} b_l(\eta)\beta_l \tag{3.13}$$

A basis for $s(\eta)$ defines the space of all functions of which $s(\eta)$ or its approximation is an element. With $b_l(\eta)$ as the l*th* basis function and β_l the l*th* parameter, substituting (3.13) into (3.12) results in a linear model (Wood, 2006) so that estimation methods for linear model such as least square method can be employed. For example, a basis for the space of cubic or less order polynomials is

$$b_1(\eta) = 1, \quad b_2(\eta) = \eta, \quad b_3(\eta) = \eta^2, \quad b_4(\eta) = \eta^3$$

in which case we have

$$s(\eta) = \beta_1 + \eta\beta_2 + \eta^2\beta_3 + \eta^3\beta_4 \tag{3.14}$$

The above representation is for a single 4^{th} degree polynomial fitted to η in its entirety. As previously noted, the natural cubic spline is the best smoothing function; in which case we have η divided into intervals and we fit a cubic to each segment. For a similar purpose, a modified representation of cubic spline function can be made. Let the knot locations be η^*, and the number

of the chosen knots be *q*, where *q* therefore represents the dimension or rank of the basis. The rather complicated bases for the cubic spline (Wood, 2006) are

$$b_1(\eta) = 1, \quad b_2(\eta) = \eta, \quad b_{j+2}(\eta) = R(\eta, \eta_j^*), \quad \text{for } j = 1.....q-2 \tag{3.15}$$

where, if we let *t* represent the *j*th knot location η_j^*,

$$\begin{aligned} R(\eta, t) = &\left[\left(t - 1/2\right)^2 - 1/12\right]\left[\left(\eta - 1/2\right)^2 - 1/12\right]/4 \\ &- \left[\left(\left|\eta - t\right| - 1/2\right)^4 - 1/2\left(\left|\eta - t\right| - 1/2\right)^2 + 7/240\right]/24 \end{aligned} \tag{3.16}$$

The result is a linear model representation of Z which then allows for model estimation by least squares:

$$\begin{aligned} z &= \mathbf{X}\beta + \varepsilon \quad \equiv \quad z = s(\eta) + \varepsilon \\ s(\eta) &= \beta_1 + \eta\beta_2 + \sum_{j=1}^{q-2} R(\eta, \eta_j^*)\beta_{j+2} \end{aligned} \tag{3.17}$$

where β is a q-vector of real valued coefficients and the *i*th row of model matrix ***X*** is

$$\mathbf{X}_i = \left[1, \eta_i, R(\eta_i, \eta_1^*), R(\eta_i, \eta_2^*), ...R(\eta_i, \eta_{q-2}^*)\right]$$

Further technical details about the cubic spline bases formulation can be found in (Wood, 2006).

3.5.2 Penalized Regression Cubic Splines

Once a basis has been chosen for each smooth in the model, next it is necessary to control the degree of smoothness. One method for doing this is to fix the basis dimension *q* (number of knots) at a slightly higher level than necessary and add a "wiggleness" penalty to the least square fitting criterion (Wood, 2006). That is, fit model to the data by minimizing

$$\|y - \mathbf{X}\beta\|^2 + \tau \int_0^1 [s''(\eta)]^2 \, d\eta \tag{3.18}$$

over all twice continuously differentiable functions $s(.)$ having integrable second derivatives. The first term in (3.18) measures the goodness-of-fit to the data and from here the wiggleness of the function arises; and without a penalty term the model becomes strictly an interpolation of q knots. The second term in (3.18) represents quantified "wiggleness" multiplied by τ . It penalizes the first term. The tradeoff between the wiggleness (how closely the data points are tracked) and smoothing (for visual pleasing and ease of interpretation) is controlled by the smoothing parameter τ which weights the wiggleness. When $s''(\eta) = 0$ a constant slope is implied, that is $s(\eta)$ is linear, in which case we have the standard least squares problem. Otherwise, when $s''(\eta) \neq 0$ (and therefore $[s''(\eta)]^2$ is positive), the slope is changing and nonlinearity is present; therefore as τ approaches infinity the penalty term also approaches infinity. Obviously the penalty term then needs to be calibrated. For example $\tau = 0$ implies an un-penalized regression estimate (Hastie & Tibshirani, 1990; Wood, 2006). Too low τ causes the model to fit the signal plus the noise; the resulting excessive tracking or extra variability will lead to poor prediction of the missing datum by the model. The idea is to choose the best value for τ that will allow a candidate additive model to maximally predict data to which it was not fitted. Fortunately there are algorithms for finding the optimal value for τ including the ordinary cross validation (OCV) and the generalized cross validation (GCV); basically both methods find $\hat{\tau}$ that minimizes the difference between the true function $s(\eta)$ and the spline estimate $\hat{s}(\eta)$:

$$CV = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{s}(\eta_i) - s(\eta_i) \right)^2$$

Since we do not know $s(\eta)$, the cross validation (CV) cannot be calculated directly, instead the expected squared error in predicting a new variable is derived as $E(CV)+\sigma^2$ and worked with in slightly different ways in the two methods; details of which can be found in (Wood, 2006). The GCV approach has computational advantages over the former; hence GCV is preferred for searching for the optimal τ, that is, selecting the degree of smoothness (Wood, 2006). Whereas the approach to model estimation in AM is by penalized least squares, the method of choice for estimation in GAM is penalized likelihood maximization which in practice is achieved by penalized iterative least squares (Wood, 2006). For detailed information about the cross-validation techniques and the model estimation methods, please refer to (Wood, 2006). In the GAM procedure according to the *mgcv* package (R Development Core Team, 2008), the effective degrees of freedom (edf) is automatically calculated as a mathematical function of τ and reported in the model output. A higher edf corresponds to greater nonlinearity.

3.5.3 Estimation in Penalized Regression Splines

Expanded details of the estimation process described in this section can be found in (Wood, 2006). Briefly the penalty term in (3.18) being linear in the parameters β can be re-expressed in a quadratic form of β (for cubic splines)

$$\tau\int_0^1\left[s''(\eta)\right]^2 dx = \beta'\mathbf{S}\beta$$
$$\begin{cases} S_{i+2,j+2} = R\left(\eta_i^*,\eta_j^*\right) \\ i,j = 1,\ldots,q-2 \end{cases} \tag{3.19}$$

where S is a matrix of known coefficients with its first two rows and columns equal to zero. It follows that fitting the model reduces to minimizing

$$\|y - \mathbf{X}\beta\|^2 + \tau\beta'\mathbf{S}\beta \tag{3.20}$$

w.r.t. β given τ. An optimal smoothing parameter τ is chosen using the method of generalized cross validation. For an additive model involving two smooth functions, penalized regression spline basis is used for each smooth function. Consider two predictors U and η with all values constrained to lie in $[0,1]$:

$$\begin{aligned} y_i &= s_1(u_i) + s_2(\eta_i) + e_i; \quad e_i \sim \text{i.i.d. } N(0,\sigma^2) \\ s_1(u) &= \delta_1 + u\delta_2 + \sum_{j=1}^{q_1-2} R\left(u, u_j^*\right)\delta_{j+2} \\ s_2(v) &= \gamma_1 + v\gamma_2 + \sum_{j=1}^{q_2-2} R\left(\eta, \eta_j^*\right)\gamma_{j+2} \end{aligned} \tag{3.21}$$

where q_1 and q_2 are the number of parameters to be estimated for the corresponding smooth function. For identification, either of δ_1 or γ_1 is set to zero. The *i*th row of the model matrix for the linear model form $y = \mathbf{X}\beta + \varepsilon$ becomes

$$\mathbf{X}_i = \left[1, u_i, R(u_i, u_1^*), R(u_i, u_2^*), ..., R(u_i, u_{q_1-2}^*), \eta_i, R(\eta_i, \eta_1^*), ..., R(\eta_i, \eta_{q_2-2}^*)\right] \tag{3.22}$$

To estimate the parameters $\beta = [\delta_1, \delta_2, ..., \delta_{q_1}, \gamma_2, ..., \gamma_{q_2}]'$, we minimize the least square objective

$$\|y - \mathbf{X}\beta\|^2 + \tau_1\beta'\mathbf{S}_1\beta + \tau_2\beta'\mathbf{S}_2\beta \tag{3.23}$$

For non-normal data the Generalized Additive Models (GAMs) are set up as penalized GLMs and the model is fitted by penalized likelihood maximization using penalized iterative least square. For an example of a model that includes both non-smoothed and smoothed terms including interaction term, let X_i^* represent the model matrix of the strictly parametric (non-smoothed)

component of the model with its associated parameters θ while the s_j are the smooth functions; we have

$$\begin{aligned} g\big(E(y)\big) &= \mathbf{X}_i^{*}\theta + s_1(\eta_i) + s_2(\eta_i, x_i) + \ldots \\ s_j(\eta_j) &= \sum_{i=1}^{q_j} \beta_{ji} b_{ji}(\eta_j) \end{aligned} \tag{3.24}$$

with *g* as the known link function. To make the model identifiable, the model matrices for each smooth term is mean- or sum-centered at zero, and the model can then be represented as

$$\begin{aligned} & g\big(E(y)\big) = \mathbf{X}\beta \\ & \begin{cases} \mathbf{X} = [X^{*} : X_1 : X_2 : \ldots] \\ \beta' = [\theta', \beta_1', \beta_2', \ldots] \end{cases} \end{aligned} \tag{3.25}$$

To suppress the wiggleness contribution from each $s_j(x_j)$ the likelihood $L(\beta)$ of the model is penalized to obtain $L_p(\beta)$:

$$L_p(\beta) = L(\beta) - \frac{1}{2}\sum_j \tau_j \beta' \mathbf{S}_j \beta \tag{3.26}$$

where the smoothing parameters τ_j control the wiggleness and are themselves estimated. For the proof and the iterative estimation process the reader is referred to (Wood, 2006).

3.6 Goodness of Fit and Model Comparison

For each regression equation in the ALV model we applied the generalized linear model (Nelder & Wedderburn, 1972) so that each regression model specification is in terms of the linear predictor $X\beta$. So the deviance is output directly by the standard GLM/GAM procedure, and is defined as

$$D = 2[\log L(\hat{\Omega}_{sat}) - \log L(\hat{\Omega})]\phi$$
$$D \sim \chi^2_{n-p}$$

where $\log L(\hat{\Omega}_{sat})$ is the maximized log likelihood of the saturated model, *n* is number of observations, *p* is number of identifiable parameters, and the scale parameter $\phi = 1$ for the Normal and Binomial distributions used in the development of the ALV model. Note that there are C columns of N-vector η generated in each EM cycle as MCMC samples (N and C are number and length of MCMC chains respectively). For *p* response variables there are *p* univariate regression models fitted for each N-vector η repeatedly across C columns, to yield a $p \times C$ matrix of each element of the model fit results. One of these elements is the deviance D directly estimated by each regression model. Then the average deviance is computed over the C columns to produce a set of average values $\{\bar{D}_1, ..., \bar{D}_p\}$ for the *p* sub-models. So there are C univariate regression model fits yielding C deviances w.r.t. each response variable. These *p* deviances are then summed for the system of regression equations to give a total deviance $\bar{D}$ which indicates the overall log likelihood of the ALV model. So, to compare nested ALV models 1 and 2 we can perform the likelihood ratio test, where with hypothesis testing based on large sample limit we have approximately

$$\bar{D}1 - \bar{D}2 \sim \chi^2_{p_2 - p_1}$$

Non-nested ALV models can also be compared on the Bayesian information criterion (BIC) and Akaike information criterion (AIC) also which we are able to compute as follows:

$$AIC = \bar{D} + 2 * p$$
$$BIC = \bar{D} + (\log n) * p$$

The better fitting of the two ALV models will produce lower values of either statistic. We considered computing the BIC and AIC also at the sub-model levels and finding the average as done for the deviance, however we believe that more simulation studies will be required specifically to investigate which approach should be better, and this should be a subject of future study.

CHAPTER 4

COMPUTATIONAL DEVELOPMENT OF ALV MODEL

4.1 Computation Steps

The proposed ALV model was developed and written entirely in R language using the R 2.8.1 statistical application (R Development Core Team, 2008). The latent η vector was simulated using a random walk Metropolis algorithm available within the Markov Chain Monte Carlo Package (MCMCpack version 0.9-4) written by Martin, Quinn, & Park (2009) in R. To simulate the random vector η we employed the R function *MCMCmetrop1* available from the MCMCpack to construct a Markov sample from user-defined conditional distribution of η, using a random walk Metropolis algorithm. For diagnostic purpose the output of the MCMC simulations is analyzed with the CODA (Convergence Diagnostics and Output Analysis) package (Plummer, Best, Cowles, & Vines, 2006) that comes with the MCMCpack.

The steps involved in the extended MCEM computations are graphically displayed in Figures 4.1a-c reflecting summaries of the equations (2.20) to (2.28) . For the *k*th MCEM iteration, a single chain Markov sample of size *C* was drawn from the conditional distribution of η for the *i*th subject in the E-step via the Metropolis algorithm (Figure 4.1b). This yields for all subjects *N* independent Markov samples stored in an N×C matrix. Each column of this matrix constitutes independent observations and was plugged into the Q function one column at a time to substitute for η, given the current (*k*th) parameter estimates. The availability of estimated N-vector η as a predictor variable then allows for the new *(k+1)*th MLE's and their standard errors

to be obtained at the M-step as direct outputs by fitting standard regression models including GAM (Figure 4.1c).

The ALV model at this stage accepts continuous indicator variables (Y's), one continuous or binary distal outcome (Z), and a two-category group or treatment variable (GRP). In addition it can also accept a cluster variable as a random effect; however in its current form he ALV model can optionally include the cluster variable in its analysis only at the final EM iteration. That is, the Additive component will switch from GAM to GAMM in the final EM iteration to accommodate the the clustering factor in the data. Technically the GAMM analysis procedure combines Linear Mixed Model (LME) with GAM within its algorithm (Wood, 2006).

Figure 4.1a ALV Algorithm Flow Chart: Overview of EM Setup

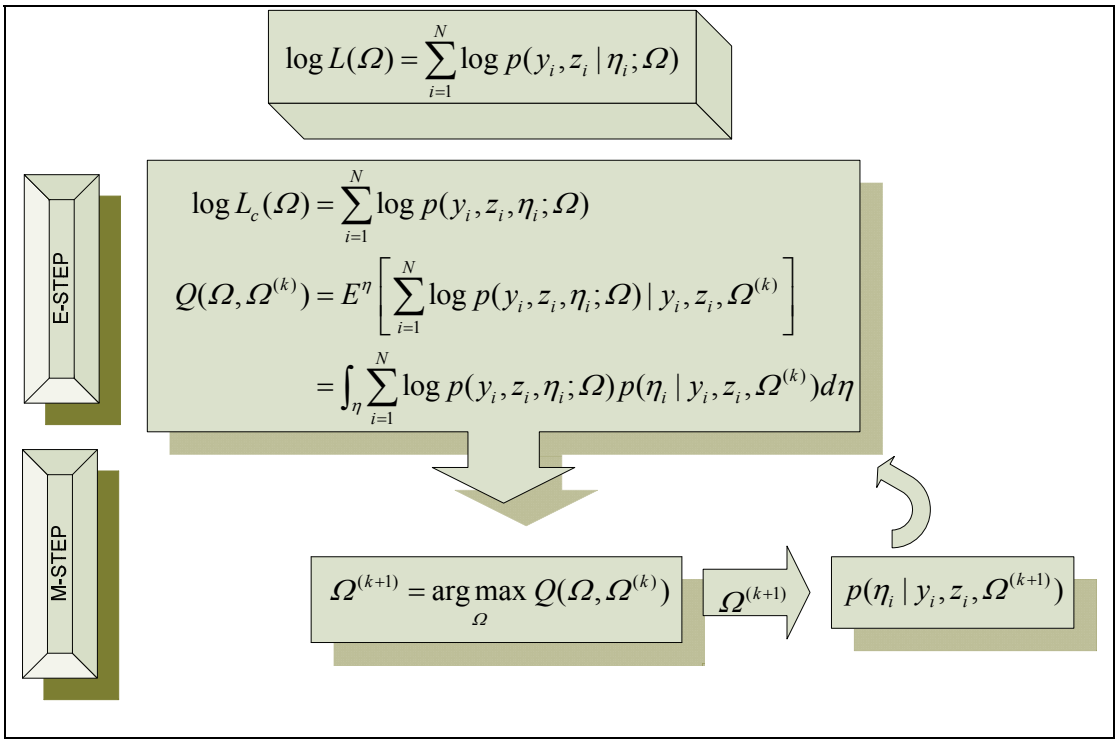


Figure 4.1b ALV Algorithm Flow Chart: Implementing the E-Step via MCMC Process

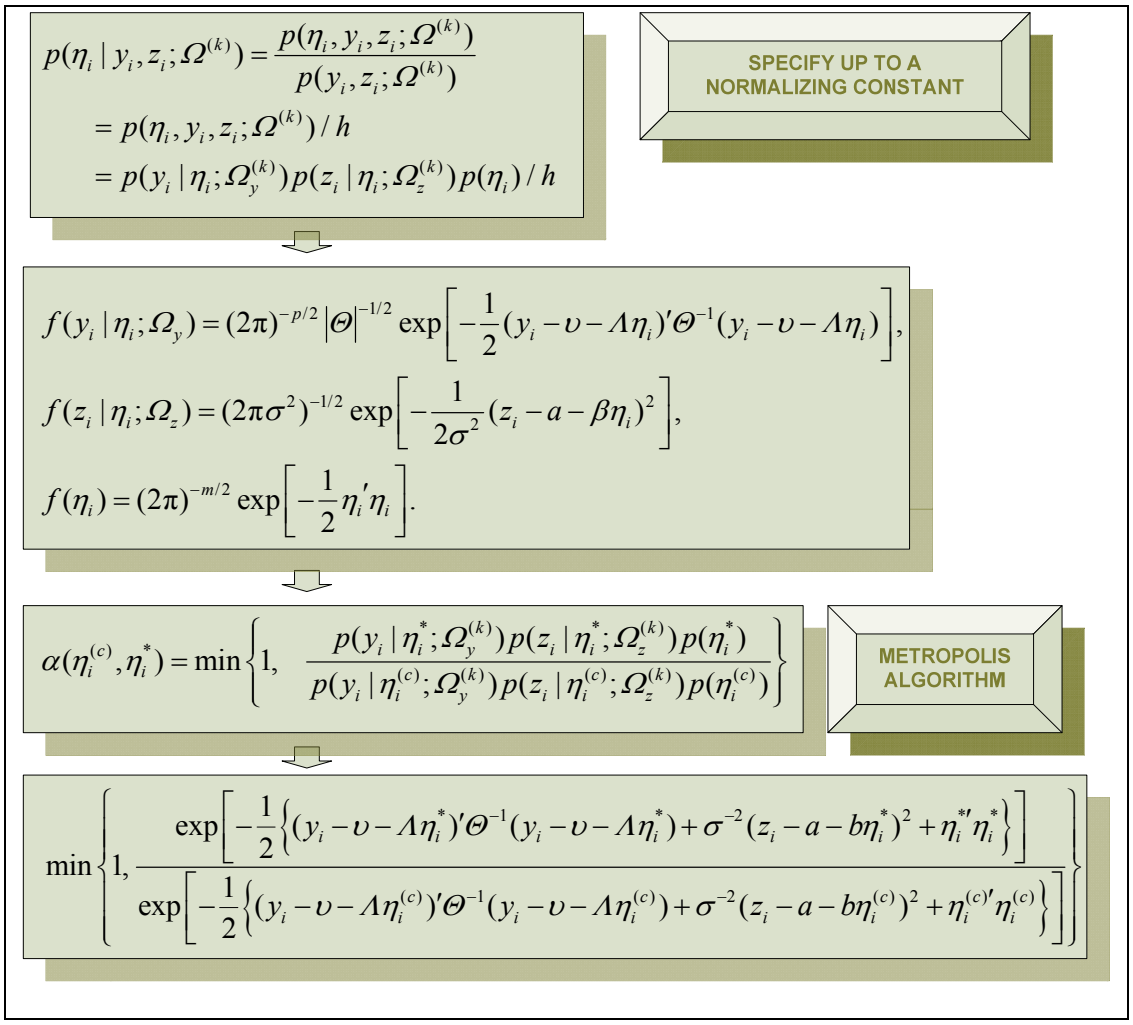
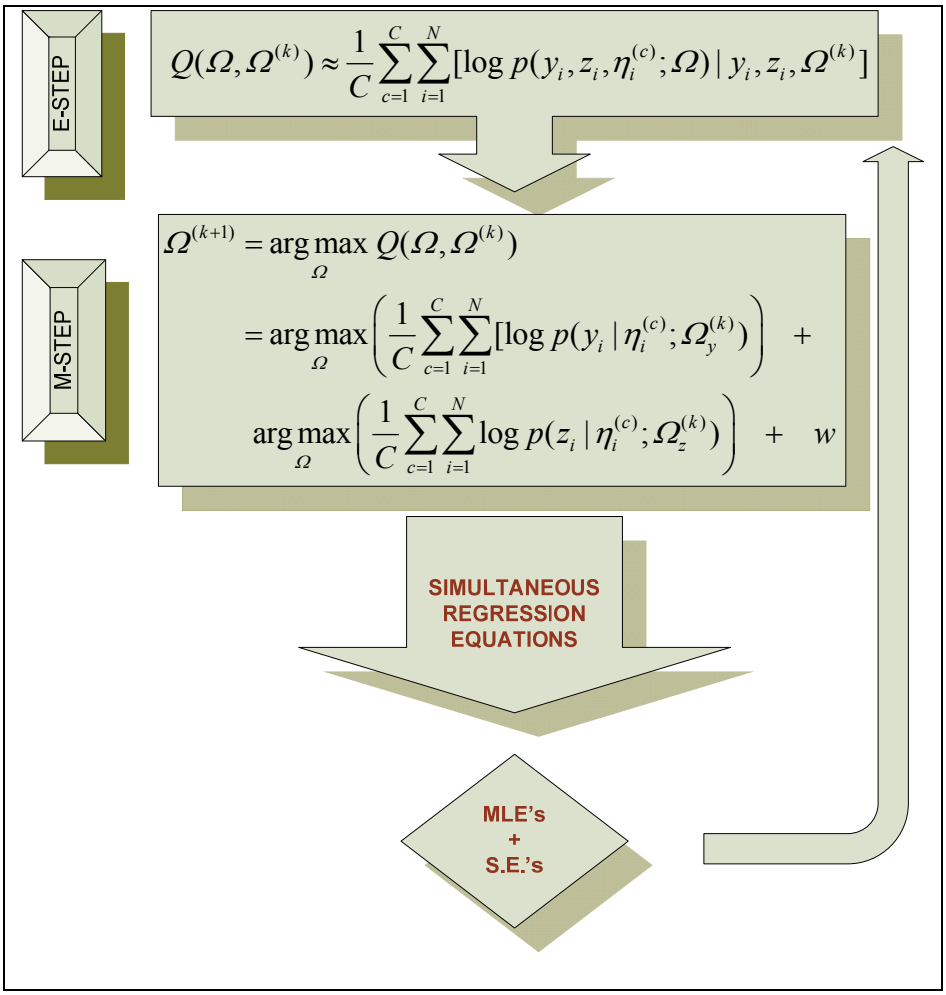


Figure 4.1c ALV Algorithm Flow Chart: Summary of the MCEM Implementation



The main parameters to be estimated require start values. We adopt the following scheme to facilitate fast convergence and efficiency of the AVL algorithm. The calculated sample means and variances of the Y's are employed as start values for the Y-intercepts υ and measurement error variances θ while Y-slopes λ are arbitrarily assigned start values, e.g. 1.0. For the GAM component, Z is regressed on GRP and Y's to obtain a start value for the Z intercept α, but the initial error variance σ^2 is also obtained from the sample variance. The above first line start

values are then used to compute the empirical Bayes' (EB) estimates to serve as start values for the N-vector η. Alternatively, a standard normal random sample can be generated as the initial η vector and we found this to work as well for our simulated data but in general this is not our preferred choice. It is important to choose start values that allow the MCMC chain to start as close to the center of the target distribution (conditional distribution of η) as possible (e.g. EB estimates, approximate MLE's) as this will greatly reduce the required burn-in time and facilitate a well mixing chain (Walsh, 2004). In a well mixing chain the entire space of the target distribution is sufficiently sampled. In a situation where the target distribution has multiple peaks, a simulated annealing approach would be an alternative for obtaining start values on a single-chain such as ours (Walsh, 2004), but our target distribution is uni-modal.

The 'pseudocode' for ALV model is as follows:

Step 1. Preliminary

Dataset: Arrange variable columns in the order $\{Y_1,...,Y_p,Z,GRP,Cluster\}$. Remove rows with missing values.

Start values –

- Parametric coefficients: supply λ_0's, compute μ_0's, θ_0's, σ_0^2's
- Generate or compute the initial vector η_0
- Nonparametric coefficients: regress Z on η_0 using the GAM function to obtain initial MLE's of α_0 and β_0's
- Use η_0 (rescaled to lie in [0,1]) to construct initial GAM model matrix $Xmat_0$

Step 2. Start EM Loop

for k = 1 **to** maximum iteration **do until convergence**

- update EM counter, parameters, η vector - to give *k*th values
- create matrices for holding results generated in *k*th iteration

Metropolis Loop (generates MCMC samples)

for i = 1 **to** N **do**

- update subject counter
- input: subject level data: *i*th row of (i) dataset (ii) η vector (iii) Xmat
- apply MCMCmetrop1R function to the input data (subject level)
- output: $N \times C$ η-matrix; the rows consist of N independent Markov chains of length C; each column is an N-vector η

end for

Regression Loop (produces MLE's)

for j = 1 **to** C **do**

- input: (i) $N \times C$ matrix consisting of columns of η (ii) dataset
- regression models are plugged into the conditional likelihood functions:
- fit linear model to each Y-indicator with *j*th column of η-matrix as a lone predictor

- fit GAM to Z (response variable) with *j*th column of η-matrix + GRP + 2-way interaction terms as predictors
- output: (*k+1)*th set of (i) mle's $\{\mu's, \lambda's, \beta's\}$ (ii) standard errors of mle's (iii) $\{\theta_0's, \sigma_0^2's\}$ computed from residuals of regression equations (iv) deviance estimate for each model fit.

end for

- A total of C regression equations are fitted per response variable to yield C estimates per parameter. Final (*k+1)*th estimates are the average of C estimates

Update Xmat

- Compute the row means of the $N \times C$ eta-matrix to yield $\bar{\eta}$ for N subjects
- Use N-vector $\bar{\eta}$ to generate the (*k+1)*th *Xmat*

Compute convergence errors. **If** convergence, **stop, else return** to step 2.

End EM Loop

4.2 Criteria for Convergence of ALV Model

Convergence issues are addressed at two levels: how to ensure that the Metropolis sampler has reached a stationary distribution; and how to diagnose convergence for the E-M iterations and ensure that the parameter estimates converge to their true values.

4.2.1 Convergence of MCMC

There are two main concerns including how to eliminate dependence on start values and how to diagnose convergence of the MCMC iterations. Also there are two schools of thought on the appropriate approach to address these concerns: generate multiple chains from different start values, or simply use one long chain because this is more robust with respect to poor start values (McLachlan & Krishnan, 2008). According to the authors, whether one uses one long Markov chain or uses multiple short chains, the diagnostic tests of convergence can still be fallible; so the focus of MCMC runs should be on the precision of estimation of the expectation(s). Therefore, considering the above and the fact that our ALV algorithm involves N number of MCMC runs per EM cycle; obviously we prefer the single chain approach. Later in our simulation studies, we will place emphasis on the precision of MCMC estimates (compared to true values) in the evaluation of convergence of the ALV model.

We took advantage of the available diagnostic tests that can be conducted within the MCMCpack (Martin, Quinn, & Park, 2009) to confirm that the Markov chain converges sufficiently close to its stationary distribution. Using the R function 'raftery.diag' we were able to calculate the effects of autocorrelation in a short pilot run of a Markov chain and use the results to determine an adequate length required for the chain to achieve a stationary state. If the estimated autocorrelation is high ('dependence factor' estimate > 5), the required length of chain will be large and this can pose computer memory challenge. The memory demand can be reduced by thining the output whereby every *n*th consecutive value after burn-in period is selected and stored for use in subsequent analysis (Walsh, 2004). The results of the Raftery diagnosis also included the estimated number of 'burn-in' iterations to be thrown away at the beginning of the Markov chain, as well as plots of the sampler run.

4.2.2 Convergence of EM

The determination of convergence in the Monte Carlo EM extension is not trivial; the usual standard approach is not suitable and non-convergence may be compounded by implementation or numerical errors (Lee, Song, & Lee, 2003; McLachlan & Krishnan, 2008). According to the authors, by approximating the expectation at the E-step with values generated from MCMC samples, a Monte Carlo error is introduced and the monotonicity property of the EM algorithm is lost. One approach to monitoring of EM convergence therefore is to plot the values of parameter estimates $\hat{\boldsymbol{\Omega}}$ against the index of iteration and conclude that convergence has taken place if the process has stabilized with random fluctuations around the estimates (Wei & Tanner, 1990; McLachlan & Krishnan, 2008). When the number of parameters to be estimated is large (as in our case) an alternative approach is to monitor changes in a function of $\hat{\boldsymbol{\Omega}}$ such as the log-likelihood function (Meng & Schiling, 1996). It is also known that the log-likelihood function can still fluctuate randomly along the EM iterates even in the absence of implementation or numerical errors (Lee, Song, & Lee, 2003), however this has been shown to be adequate for the purpose of statistical inference (Lee & Zhu, 2002; Meng & Schiling, 1996). Although we included both of these methods in our approach, we placed relatively more emphasis the monitoring of log-likelihood function.

A special method is required to monitor convergence of a likelihood function in the EM setting. We would be interested specifically in the change in observed data log-likelihoods between two consecutive EM iterations (*k, k+1*), which can be obtained from the logarithm of the ratio of the two likelihood values (logLR):

$$\log \mathrm{LR}(\boldsymbol{\Omega}^{(\mathbf{k}+1)}, \boldsymbol{\Omega}^{(\mathbf{k})}) = \log \frac{\mathrm{p}(\mathbf{y}, \mathrm{z}|\eta; \boldsymbol{\Omega}^{(\mathbf{k}+1)})}{\mathrm{p}(\mathbf{y}, \mathrm{z}|\eta; \boldsymbol{\Omega}^{(\mathbf{k})})} \tag{4.1}$$

Ideally the ratio will be easy to evaluate if using marginal likelihood after integrating out η. However, similar to the experience of Lee, Song, & Lee (2003) the observed likelihood in our case is difficult to obtain analytically, so we follow the authors' approach (bridge sampling method) by applying the Meng and Schiling's approximation formula (Meng & Schiling, 1996) based on the complete data likelihood with respect to c*th* MCMC iterate within the *k*th EM cycle as follows:

$$\widehat{\log \mathrm{LR}}(\boldsymbol{\Omega}^{(\mathbf{k}+1)}, \boldsymbol{\Omega}^{(\mathbf{k})}) = \log\left\{\sum_{\mathbf{c}=1}^{\mathbf{C}}\left(\frac{\mathrm{p}(\mathbf{y}, \mathrm{z}, \eta^{\mathrm{k},(\mathrm{c})} \mid \boldsymbol{\Omega}^{(\mathbf{k}+1)})}{\mathrm{p}(\mathbf{y}, \mathrm{z}, \eta^{\mathrm{k},(\mathrm{c})} \mid \boldsymbol{\Omega}^{(\mathbf{k})})}\right)^{1/2}\right\} - \log\left\{\sum_{\mathbf{c}=1}^{\mathbf{C}}\left(\frac{\mathrm{p}(\mathbf{y}, \mathrm{z}, \eta^{\mathrm{k}+1,(\mathrm{c})} \mid \boldsymbol{\Omega}^{(\mathbf{k})})}{\mathrm{p}(\mathbf{y}, \mathrm{z}, \eta^{\mathrm{k}+1,(\mathrm{c})} \mid \boldsymbol{\Omega}^{(\mathbf{k}+1)})}\right)^{1/2}\right\} \tag{4.2}$$

where $\{\eta^{\mathrm{k}(\mathrm{c})}, \mathrm{c} = 1, \ldots \mathrm{C}\}$ are simulated from the conditional distribution of η evaluated at *k*th estimates. The aim is to claim approximate convergence when the change in consecutive likelihoods along EM iterations becomes very small and fluctuates within a desired level, that is, the log of the likelihood ratio fluctuates near zero. The approximation is claimed to be sufficient for the purpose of statistical inference (Meng & Schiling, 1996; Lee & Zhu, 2002). Similar to the monitoring of parameter estimates, approximate convergence is assumed when the estimated logLR approaches and fluctuates in the neighborhood of zero.

Most importantly, once within the region of such steady fluctuation, it is necessary to come up with appropriate values of parameter estimates at convergence. Some authors obtained the average of all values within the region for the final estimates with respect to individual parameters, while some selected the parameters values at arbitrary *k*th iteration within the region as the MLE's (Lee, Song, & Lee, 2003). In our case, regression model deviances are also available as output using our algorithm. So, in order to establish a more objective criterion for

point convergence, we decided to also monitor the ALV model deviance computed as the sum of deviances for all regression model fits comprising the ALV model. So, from within the steady fluctuation region we choose the parameter values corresponding to the point of minimum deviance across the EM iterations i.e. set of parameters that provide the best overall model fit to the data. So our strategy for deciding EM convergence is to first monitor the log of the likelihood ratio to ensure the region of stable fluctuation around zero is achieved, then choose parameter estimates associated with minimal model deviance within the stable region.

CHAPTER 5

APPLICATION TO SIMULATED DATA

5.1 Simulation Design and Background Information

A simulation study was carried out to evaluate the properties of the estimation procedure of the proposed ALV model. The simulation data structure mimics a randomized control trial investigating the variation in impact of treatment G on an outcome Z across the levels of a baseline risk (η); with the additional challenge that η is unobservable and must be inferred from five observed variables Y1 to Y5. The Y's are assumed to be continuous and multivariate Normal, G has two levels, and Z may be binary or continuous with a normal distribution. Also, the Y's and Z are conditionally independent given η . The ALV model in this case consists of six regression sub-models corresponding to five Y's and one Z. We used different specifications, each with 50 replicated datasets, and repeated for each of three sample sizes N = 100, 200, 300 (see Table 5.1). All simulation datasets were generated in R 2.81 (R Development Core Team, 2008). Complementary analyses were performed in Mplus version 5.1 (Muthen & Muthen, 2008).

The simulations were used to assess two major important questions for the model: (1) what is the long term behavior of the ALV model (pattern of convergence)? (2) How well does the ALV model perform under (a) different study sample sizes and (b) different functional forms of the relationship of Z to η ? To answer the posed questions we performed simulations under 12 different scenarios constructed from the combinations of the following data structures (see Table 5.1): Z-scale (continuous and binary), η -effects (linear and nonlinear), and sample size (100,

200, 300). Under each scenario we investigated (i) the optimality requirements for the MCMC sampler, (ii) the ALV model convergence characteristics with a single run of the model through 100 EM iterations, and (iii) the accuracy of the ALV model estimates and their standard errors by running 50 replications. Regarding the MCMC sampler optimality conditions, we pilot tested the MCMC sampling to (a) fine tune the variance of proposal distribution so as to achieve a Metropolis sampling acceptance rate of between 0.43 and 0.47, (b) determine the shortest length of Markov chain required to achieve stationarity (burn-ins), (c) assess variance inflation factor I of the data, if $I \geq 4$ then determine how much thinning is required to reduce the autocorrelation in the data to a minimum, (d) determine the optimal size of MCMC samples that is required to estimate η accurately and efficiently, i.e. the minimum size that is adequate for the purpose of statistical inference.

Table 5.1 Twelve Simulation Scenarios used to Assess ALV Model Performance

		Sample size		
Scale of Z	Z dependence on η and η*Z	100	200	300
Continuous	Linear	1	2	3
	Nonlinear	4	5	6
Binary (logit)	Linear	7	8	9
	Nonlinear	10	11	12

If we can assume a joint multivariate normality for all dependent variables (Y's and Z) the ALV model analysis will mimic a simple linear CFA. However, unlike the conventional CFA, in addition to solving linear equations, the ALV model is also able to model complex and unknown relationships in the Z sub-model component. So the major difference between a simple linear CFA and the proposed ALV model in its current formulation is found in the functional form of the regression sub-model of Z (the GAM component). Therefore for brevity we illustrate the results of our simulation study with the report on six representative scenarios that capture the span of ALV performance under two standard conditions (simplest & most complex) based on the functional form of the Z regression equation. The two functional forms include a continuous Z linearly related to both η and $\eta * G$ interaction term (1st row of Table 5.1), and a binary Z related to both predictor terms in a nonlinear fashion (last row of Table 5.1), and are presented as schemes 1 & 2. The first standard condition (1st row, scheme 1) allows for the assessment of ALV performance when joint multivariate normality can be assumed for the data (Y's and Z conditioned on η), equivalent to a standard linear CFA. Importantly, the ALV model results in this case can be compared to the results of CFA performed by a standard statistical application such as Mplus. The second simulated condition (last row, scheme 2) enables us to evaluate the ALV model application to more complex data. Such condition includes when Z has a binary distribution conditioning on η, plus the presence of a complex nonlinear dependence of Z on the Y's indirectly through η. The dependence is not fully specified except that the variables are conditionally independent. The emphasis in the latter evaluation is therefore on how well the ALV model is able to recover the true η and uncover the functional forms used to generate the data.

For all simulations, the measurement sub-model component of the ALV model connecting the latent η and the five observed indicators $\mathbf{y_i}$, i=1,...N is given by

$$\mathbf{y_i} = \boldsymbol{\upsilon} + \boldsymbol{\Lambda}\eta_i + \boldsymbol{\varepsilon_i}; \quad \eta_i \sim N(0,1).$$

Population values for this component are assigned as $\boldsymbol{\upsilon} = \{\upsilon_1,...,\upsilon_5\}' = 0$, $\boldsymbol{\Lambda} = \{\lambda_1,...,\lambda_5\}' = 1$, and $\boldsymbol{\Theta} = \text{diag}\{\theta_1,...,\theta_5\}' = 0.5$. A standard normal N-vector η was generated first, then the response variables Y's and Z were generated conditioned on η. While a linear model is specified for each Y, both linear and nonlinear regression models of Z (second component of the ALV model) were specified. Nonlinearity is described by inclusion of appropriate higher degree polynomial terms in the model. Let n = N/2 where N is the number of subjects in the sample; separate specification for each treatment group G is as follows:

$$\begin{aligned}
&\text{Scheme 1 (Linear)}\begin{cases} G=0: & z_{i=1:n} = \beta_{00} + \beta_{10}\eta_{i=1:n} + e_{i=1:n} \\ G=1: & z_{i=(n+1):N} = \beta_{01} + \beta_{11}\eta_{i=(n+1):N} + e_{i=(n+1):N} \end{cases}; \quad e_i \sim N(0,\sigma^2) \\
&\text{Scheme 2 (Nonlinear)}\begin{cases} G=0: & z_{i=1:n} = \beta_{00} + \beta_{10}\eta_{i=1:n} + \beta_{20}\eta^2_{i=1:n} + \beta_{30}\eta^3_{i=1:n} \\ G=1: & z_{i=(n+1):N} = \beta_{01} + \beta_{11}\eta_{i=(n+1):N} + \beta_{21}\eta^2_{i=(n+1):N} + \beta_{31}\eta^3_{i=(n+1):N} \end{cases}
\end{aligned} \tag{5.1}$$

For the second scheme we then simulated binary Z^* to have probability prob(z=1)=1/[1+exp(-z)] where *z* is probability on the logit scale. Model specifications for scheme 1 were $\beta_{00}=\beta_{01}=0$, $\beta_{10}=0.2$, $\beta_{11}=0.7$, and $\sigma=0.5$. For the second scheme we specified $\beta_{00}=1, \beta_{10}=-1$, $\beta_{20}=-0.5, \beta_{30}=1$ for group 0; $\beta_{01}=0$, $\beta_{11}=-3$, $\beta_{21}=0.4$, $\beta_{31}=0.6$ for group 1.

5.2 Monitoring Convergence

5.2.1 Convergence Pattern: MCMC loop

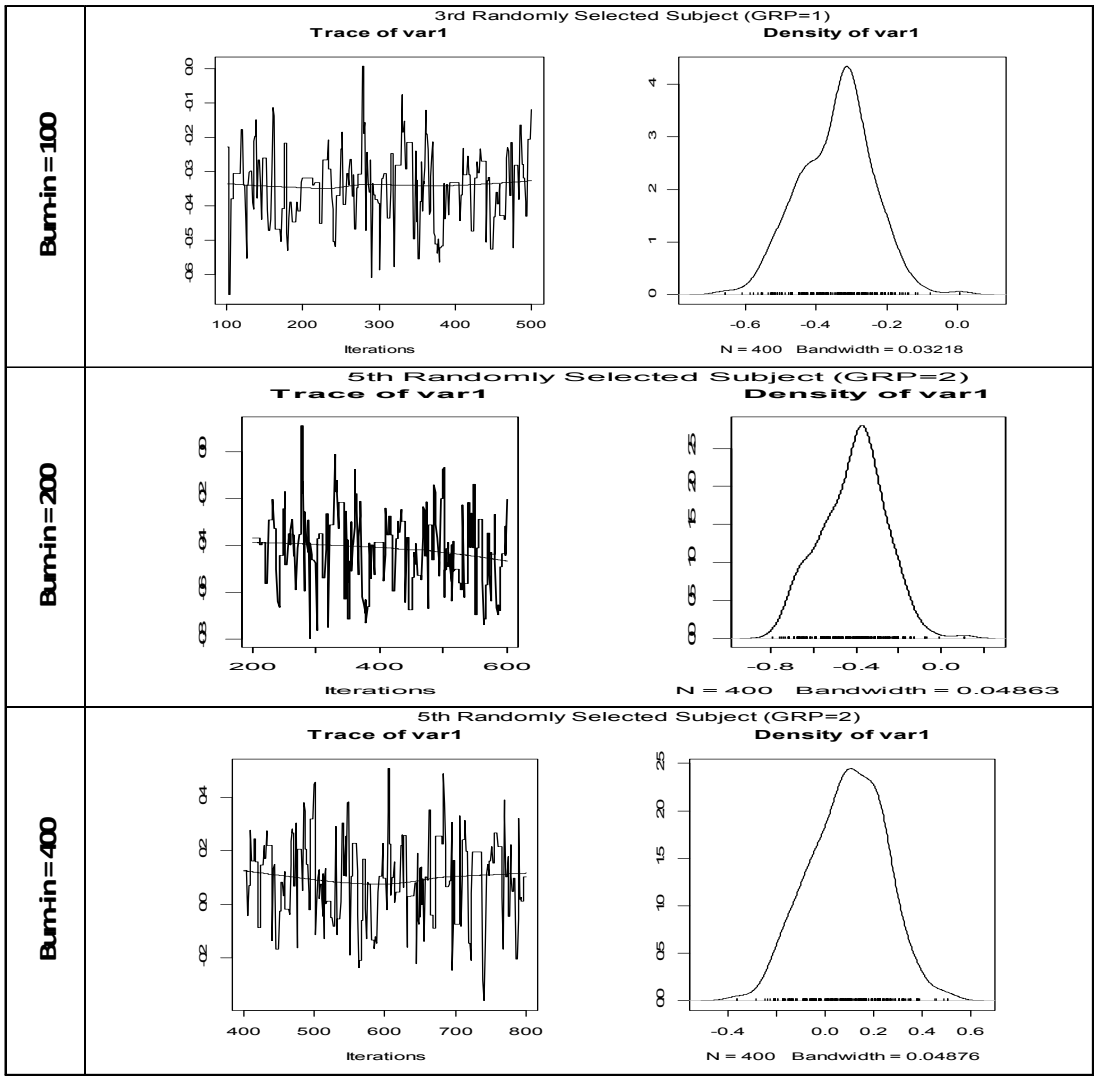
The studies of the MCMC convergence were carried out at the subject level where the conditional distribution of η is randomly generated from an inner loop inside an EM iteration cycle. Guided by the Raftery diagnostic tests results from several runs in which we looked at the times series trace of across number of MCMC iterations, we found that a relatively shorter MCMC chain with 400 iterations after a burn-in period of at least 100 iterations is generally sufficient for the purpose of inference with the ALV model. For all our simulated data the calculated sample inflation factor due to autocorrelation was generally low at about 0.4 (less than 0.5) (Chib & Greenberg, 1995), and we found that thinning of the MCMC samples did not change our results in any significant way. The acceptance rates for the Metropolis algorithm ranged between 0.40 and 0.47 (Lee & Zhu, 2002; Gamerman & Lopes, 2006).

For a snapshot illustration the results of three Metropolis sampling runs are shown in Figure 5.1; the purpose here is to compare the graphical outputs of the Raftery test for three different burn-in periods. The dataset used was generated according to scheme 1, and the results for the subject sample size N =300 is reported here. Potentially N trace/density plots could be generated in each EM cycle for all the subjects in the sample. However each run producing a trace plot in the Figure 5.1 occurred in the *k*th EM cycle and was carried out on the *i*th subject randomly selected from the subjects sample stratified by treatment group. Each plot in the left column depicts a trace of accepted η_i values across 400 random-walk Metropolis samplings. Note that by default in R (difficult to override) the 'N' in the density plot label (right panels) represents MCMC sample size (for the *i*th subject) and not subject sample size; this confusion with use of symbols arises in this instance only. So each density plot depicts the distribution of

MCMC samplings from a single run for the *i*th subject in the *k*th EM cycle; under different burn-ins. For the trace plots (left panels), the horizontal scale starts from the *(b+1)*th MCMC iteration after a burn-in period of length *b*. Any long flat segment of the trajectory corresponds to iterations where all proposed η values were being rejected; this is not desirable. The presence of multiple vertical spikes indicates well explored sampling space. We want the Markov chain to be 'well mixing' and this is achieved when 'the time series looks like white noise' (Walsh, 2004). In addition, if stationarity has been reached the average value of η across the iterations should be approximately linear and horizontal; if it appears to be drifting, it may suggest inadequate burn-in period.

The results reported here in Figure 5.1 are representative of our findings for several subjects with different sample sizes under the different model specifications we tested. They show a fair settling of the traces (linear trend) with good mixing produced with the three choices of burn-in, therefore we found the shortest burn-in period of 100 to be most efficient. In addition, apart from the relatively greater computation time and memory demand by the longer burn-in periods, we did not see any noteworthy difference in the model estimates under burn-in periods of 100 or more.

Figure 5.1 Trace and Density Plots of Markov Samples for Individual Subjects (Scheme 1)



5.2.2 Convergence Pattern: EM loop

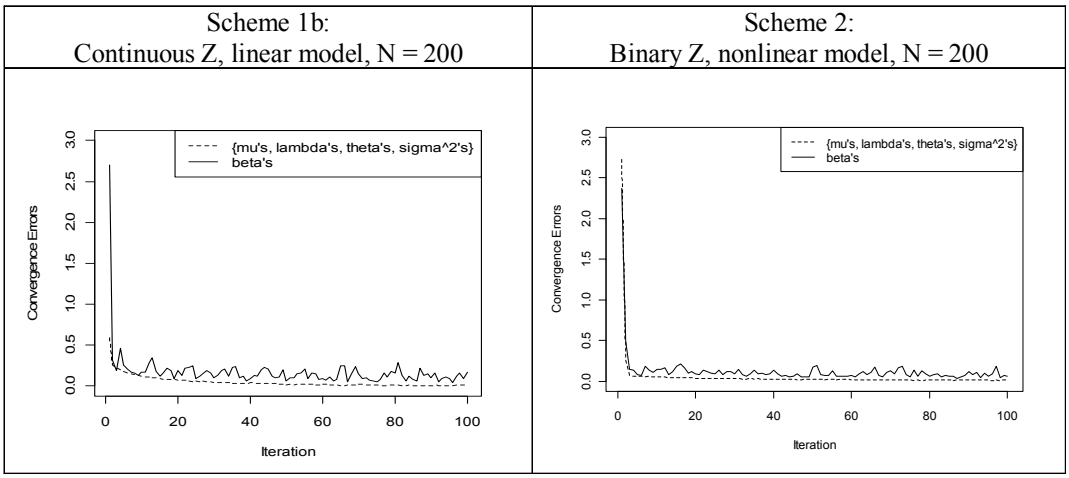
We report here the results of our investigation of the long term behavior of the ALV model estimation process with applications to simulated data. For each ALV model run we monitored across the EM iterations (i) convergence errors, (ii) approximate log of observed

likelihood ratio derived by bridge sampling, (iii) total ALV model deviance, and (iv) parameter estimates. We calculated convergence errors separately for two sets of parameters, the smoothing coefficients and the remainder parametric ML estimates. To compute the convergence error for the current EM iteration given P parameters we apply the formula:

$$\text{new error} = \sqrt{\sum_{p=1}^{P}\left(\text{old estimate}_p - \text{new estimate}_p\right)^2} \tag{5.2}$$

Two representative plots of sequences of convergence errors across 100 EM iterations are displayed in Figure 5.2. The two convergence error curves in either plot (dashed line for the parametric set of estimates and solid line for the smoothing spline coefficients) show dramatic drop before flattening out. The steady portion of each trajectory is fairly linear for the parametric set but values of the smoothing coefficients show random fluctuation within a small range. Note that the starting convergence error for the parametric set is relatively small for scheme 1 that corresponds to linear CFA analysis. This is because we started very close to the true values of η vector by using the empirical Bayes' estimates of η as start values in the Metropolis algorithm. However such approximation of η is less accurate in scheme 2 where multivariate normality does not hold, therefore the corresponding starting convergence error in this particular case is expectedly higher. The patterns are otherwise rather similar.

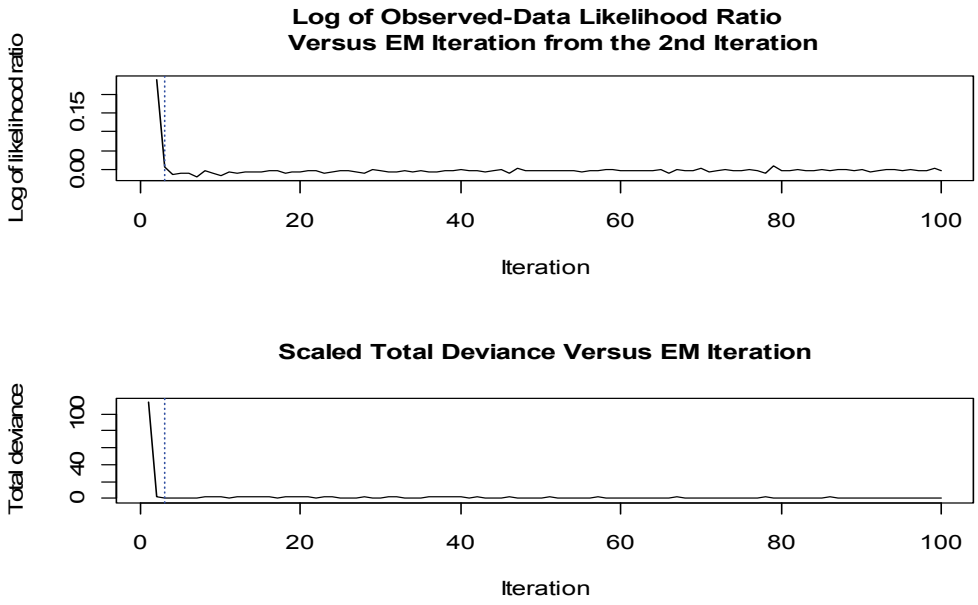
Figure 5.2 Convergence Errors versus EM Iteration



In the same ALV model run (scheme 2), the consecutive values of log of likelihood ratios and the summed deviances from all six regression sub-model estimations were plotted against the index of EM iterations (Figure 5.3). From the top graph we see that the log of likelihood ratios curve quickly approaches zero and thereafter continues to fluctuate within a narrow band around zero. This pattern is consistent with reports of previous similar studies in which the authors used the Monte Carlo EM (MCEM) approach (Lee & Song, 2007; Lee & Zhu, 2000; McLachlan & Krishnan, 2008). In the bottom graph of Figure 5.3 the total deviance scores had been rescaled so that the minimum value equals zero. The deviance curve reaches a minimum in the 4th iteration (vertical dashed line) before stabilizing; based on our criteria for convergence we concluded convergence at this point. A trajectory with an early 'pit' followed be steadiness has been the typical finding from all our simulations results describing the trace of ALV model deviance. Therefore we believe that the bottom of the 'pit' likely represents a global minimum on the trajectory. Although the linear model structure probably indicates this is the case, we cannot

assume this in general. For efficiency, once convergence is decided at this minimum model deviance, the ALV algorithm is terminated at a couple of EM iterations afterwards.

Figure 5.3 Scheme 2: Binary Z, Nonlinear Model, N=200



Just for completeness we also monitored the individual parameters in the parametric set of estimates just to explore how these parameters behave as the iterations in EM increase; some results are displayed in Figures 5.4 and 5.5 where start values on the Y-axis correspond to zero iteration. From these Figures, the residuals (thetas, sigma^2) and z-intercept (GAM component) stabilize rather quickly by the 4th iteration, our decided point of convergence; however the y's and lambdas (measurement intercepts and slopes) show gentle steady increase and only start to stabilize as from around the 100th EM iteration.

Again, from all of our simulation the results the model deviance typically becomes stable as from around 10th to 20th iteration after the minimum deviance has already been achieved;

similar patterns are exhibited by the model residuals. Therefore we suspect that the drifting values of the y-intercepts and lambdas indicate possible multiple solutions; this may need to be explored further in future studies. We believe these findings are further justifications for our approach of choosing solutions at the point of minimum model deviance as these solutions will be better than if chosen at any other point in the iterations. For these reasons, we did not see the need to extend the EM runs beyond 100 iterations just to show where the y-intercepts and lambdas finally stabilize.

Figure 5.4. Sequences of Parameters (Scheme 1) Across 100 EM Iterations

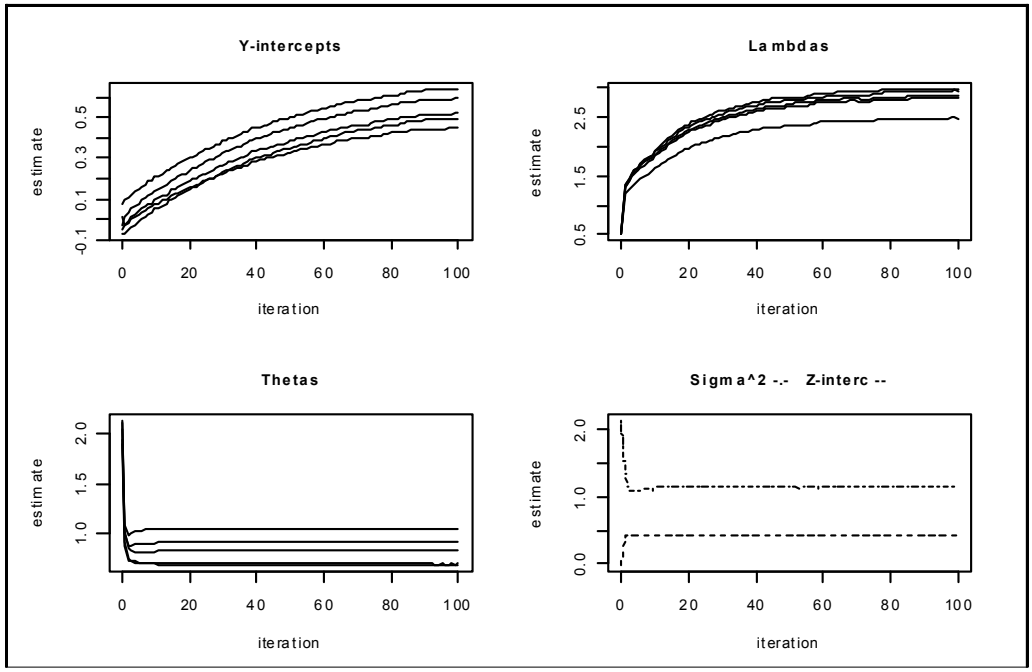
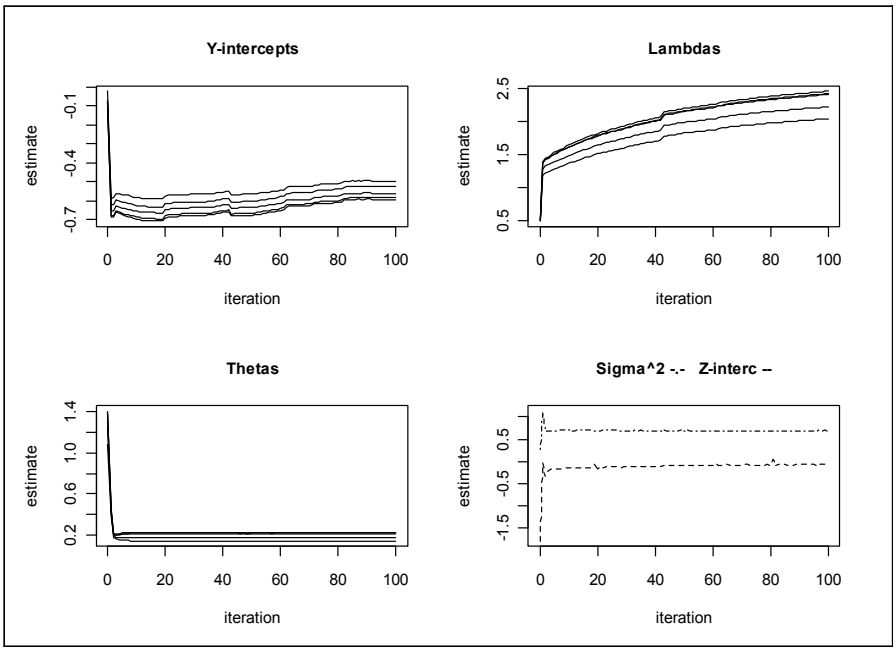


Figure 5.5. Sequences of Parameters (Scheme 2) Across 100 EM Iterations



5.3 Assessing Performance of ALV Model

We performed 50 replications of the ALV model analysis using datasets of different sample sizes (N = 100, 200, 300) generated according to the scheme 1 where joint multivariate normality is assumed for the {Y's, Z}. For the Monte Carlo part of each replication, we used burn-in =100, MCMC sample = 400. To be conservative we stopped the ALV algorithm at two iterations subsequent to reaching the minimum point on the total deviance curve. This is based on our consistent findings of an early convex shape (pit) before a prolonged flat trajectory for all the plots of deviance against EM iterations in our simulations for studying ALV model convergence (see section 5.2.2).

To evaluate the overall accuracy of the ALV model we calculated the following summary statistics for each parameter estimate based on 50 replications:

Bias– This was calculated as the difference between the true value and the computed mean of estimates.

Standard deviation (SD) – This is the empirical standard deviation of the parameter estimates across replications.

Root mean square error (RMSE) – This was calculated as the square root of the sum of the variance (of the estimates across replications) and the squared bias.

Standard error average (SE) – This is the mean of the standard errors estimated by ALV model for each parameter estimate across the replications.

SE/SD – This ratio was used to assess the accuracy of the standard errors estimated by the ALV model. If the number of replications is sufficiently large, the empirical SD can be taken as the standard error of estimate. Therefore, assuming we have sufficient number of replications, correctly estimated SE should closely approximate the empirical SD. However given the extensive computations involve in our simulations we have arbitrarily limited our replications to 50.

5.3.1 Performance under Scheme 1

The replication study based on linear models (scheme 1) helps establish that the ALV algorithm was set up correctly; and the results are reported in Table 5.2. Overall, the estimates produced by the ALV model are close to their true values as evidenced by the very small RMSE's, and the values further reduce (i.e. the performance improves) as the sample size

increases. However, the residual variance estimates (theta's, sigma^2) show little change across the different sample sizes, possibly masked by round-off errors. In addition, the estimated residual variances are considerably smaller than the specified values for the error terms ('true values') used in generating the data, hence the high values of recorded biases. Alternatively the true values of the residual variances may be approximated by replicating OLS regression equations with true eta as predictor, but we decided that this is not crucial to our study. While the bias associated with the y-intercept estimates declines as sample size increases, no clear pattern is seen with respect to the estimated slopes (lambdas). The recorded bias in z-intercept estimation appears not to be influenced by sample size. Also, while the SE/SD columns show values close to 1 for the intercepts and thetas, the values recorded for the lambdas are small. This indicates potential bias (or possibly imprecision due to insufficient number of replications) in the ALV model estimation of standard errors of estimates for the lambda parameters specifically, although there is improvement as sample size increases.

Based on our simulation findings above we believe that the measurement part of the ALV model may not yield unique solutions to the parameter estimates (intercepts and lambdas) under the current stopping rule we have adopted for convergence. As previously mentioned in earlier section, the potential existence of multiple solutions may be reflected in the delayed stabilization seen for the Y-intercepts and lambdas long after the thetas, Z-intercept and sigma^2 have stabilized (see Figures 5.4 & 5.5). Based on our stopping rule, convergence is diagnosed at the point of minimum deviance on the condition that the approximate observed log-likelihood ratio has stabilized (is fluctuating around zero) (see Figure 5.3), even when the Y-intercepts and lambdas are yet to. Although thereafter the stable sequence of the model deviance did not change considerably from the minimum, it is most efficient to stop the algorithm soon after the minimum is crossed. We believe that running the model longer than is allowed by our stopping rule will not

yield improvement in the estimation of the latent variable η which is our major focus, however further studies are needed in this area.

In addition to the above replication study of ALV model performance we also compared its analysis results to those obtained from a standard reference statistical application such as Mplus. Both statistical methods were applied however to only one copy of the simulated datasets (scheme 1, N = 300), in which a simple confirmatory factor analysis (CFA) was performed in Mplus. Although no definitive conclusion can be drawn from the results based on a single replication, the following comparison analyses offer a glimpse into some other aspects of the performance of ALV model. We found that the results of both analyses (see Table 5.3) are similar; although relatively smaller standard errors are recorded for the ALV model, the residual variance estimates from both models are close.

Next we used the results of the same set of analyses (one replication, N=300) to make comparisons between (1) the true η, (2) Empirical Bayes (EB) estimates of $\hat{\eta}$ obtained from Mplus output, and (3) MCMC estimated $\hat{\eta}$ from the ALV model. As revealed in Figures 5.6 and 5.7, the ALV model estimated $\hat{\eta}$ is nearly identical in distribution to both true η and EB estimates. This suggests that the ALV algorithm is able to accurately estimate the latent η (conditional distribution) underlying the outcome variables Y's and Z in the data. These results based on a single dataset are only preliminary; the accuracy of ALV model in estimating the latent factor will be examined further with replication studies later under scheme 2.

Table 5.2 ALV Model Estimation Performance under Scheme 1 (50 Replications)

Conditional on η, Y's are Linearly Related to Z

		N = 100			N = 200			N = 300		
Parameter	Pop	Bias	SE/SD	RMSE	Bias	SE/SD	RMSE	Bias	SE/SD	RMSE
Y1-intercept	0	-0.161	0.906	0.026	-0.085	0.982	0.007	0.012	1.029	0.000
Y2-intercept	0	-0.159	0.896	0.026	-0.088	1.071	0.008	0.010	1.066	0.000
Y3-intercept	0	-0.160	1.003	0.026	-0.090	0.816	0.008	0.011	1.017	0.000
Y4-intercept	0	-0.164	1.276	0.027	-0.090	1.198	0.008	0.011	1.019	0.000
Y5-intercept	0	-0.160	1.050	0.026	-0.090	0.934	0.008	0.013	1.094	0.000
lambda1	1	0.076	0.493	0.007	-0.118	0.649	0.014	0.091	0.720	0.008
lambda2	1	0.084	0.518	0.009	-0.119	0.571	0.014	0.090	0.613	0.008
lambda3	1	0.085	0.532	0.009	-0.118	0.604	0.014	0.091	0.651	0.009
lambda4	1	0.078	0.510	0.008	-0.116	0.474	0.014	0.093	0.657	0.009
lambda5	1	0.083	0.487	0.009	-0.115	0.576	0.014	0.093	0.625	0.009
theta1	0.25*	-0.218	NA	0.047	-0.218	NA	0.047	-0.217	NA	0.047
theta2	0.25*	-0.216	NA	0.047	-0.217	NA	0.047	-0.217	NA	0.047
theta3	0.25*	-0.218	NA	0.048	-0.217	NA	0.047	-0.217	NA	0.047
theta4	0.25*	-0.216	NA	0.047	-0.216	NA	0.047	-0.217	NA	0.047
theta5	0.25*	-0.217	NA	0.047	-0.218	NA	0.047	-0.217	NA	0.047
Z-intercept	0	0.002	1.064	0.000	-0.002	1.188	0.000	0.032	1.009	0.001
sigma^2	0.25*	-0.215	NA	0.046	-0.211	NA	0.045	-0.210	NA	0.044

* Variance of error term used in simulation

Table 5.3 Results of ALV and Mplus Analyses of a Single Dataset (Scheme 1, N=300)

Conditional on η, Y's are Linearly Related to Z

	MPLUS ANALYSIS		ALV ANALYSIS	
Parameter	Estim	S.E.	Estim	S.E.
Y1-intercept	-0.007	0.061	-0.014	0.010
Y2-intercept	0.034	0.060	0.027	0.011
Y3-intercept	0.000	0.060	-0.007	0.012
Y4-intercept	0.009	0.059	0.002	0.010
Y5-intercept	0.029	0.059	0.023	0.010
lambda1	1.031	0.047	1.104	0.011
lambda2	1.023	0.047	1.096	0.011
lambda3	1.014	0.046	1.086	0.012
lambda4	1.011	0.046	1.082	0.011
lambda5	1.001	0.046	1.072	0.011
theta1	0.038	0.004	0.031	NA
theta2	0.040	0.004	0.033	NA
theta3	0.047	0.005	0.040	NA
theta4	0.038	0.004	0.031	NA
theta5	0.035	0.004	0.029	NA
Z-intercept	-0.037	0.019	0.008	0.012

Figure 5.6 Boxplots of Eta Produced from Three Sources (Scheme 1, N = 300)

Conditioned on η, Y's are Linearly Related to Z

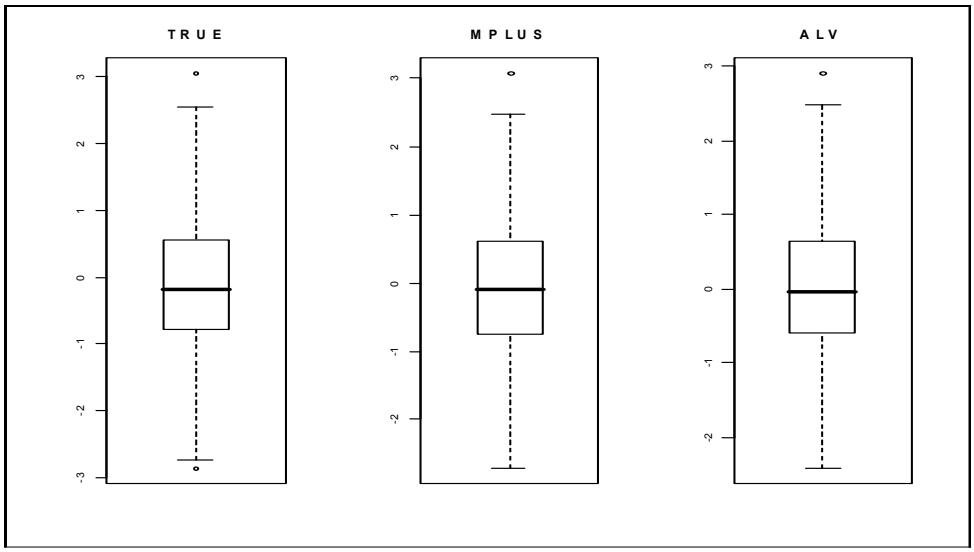
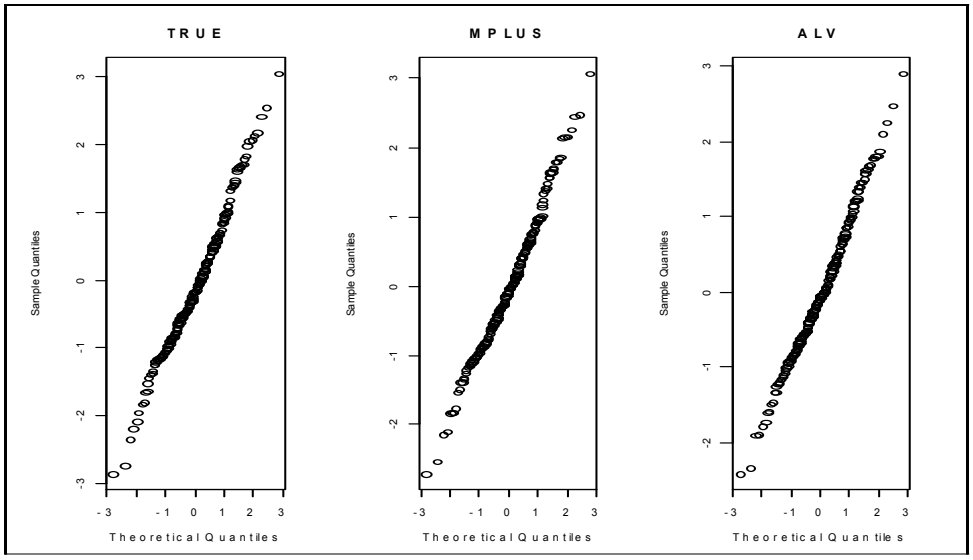


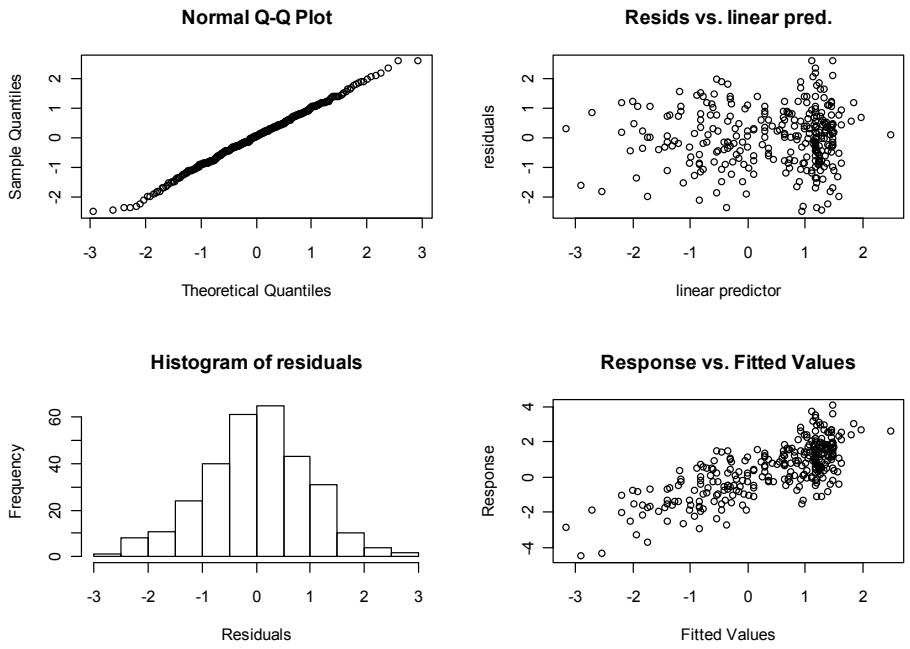
Figure 5.7 Q-Normal Plots of Eta Produced from Three Sources (Scheme 1, N = 300)

Conditional on η, Y's are Linearly Related to Z



To assess the quality of the fitting process of the GAM component of the ALV model, some basic residual plots were produced (Figure 5.8) using the *gam.check* routine in R (Wood, 2006). The closeness of the Q-Q plot to a straight line validates the Gaussian assumption for the model and the histogram of the residuals is consistent with normality. The plot of residuals versus linear predictors (top right) shows that the assumption of constant variance as the mean increases is not violated. The bottom right plot shows a positive linear correlation between the response and fitted values.

Figure 5.8 Model Checking Plots: GAM Component of ALV Model (Scheme 1, N = 300) Conditional on η, Y's are Linearly Related to Z

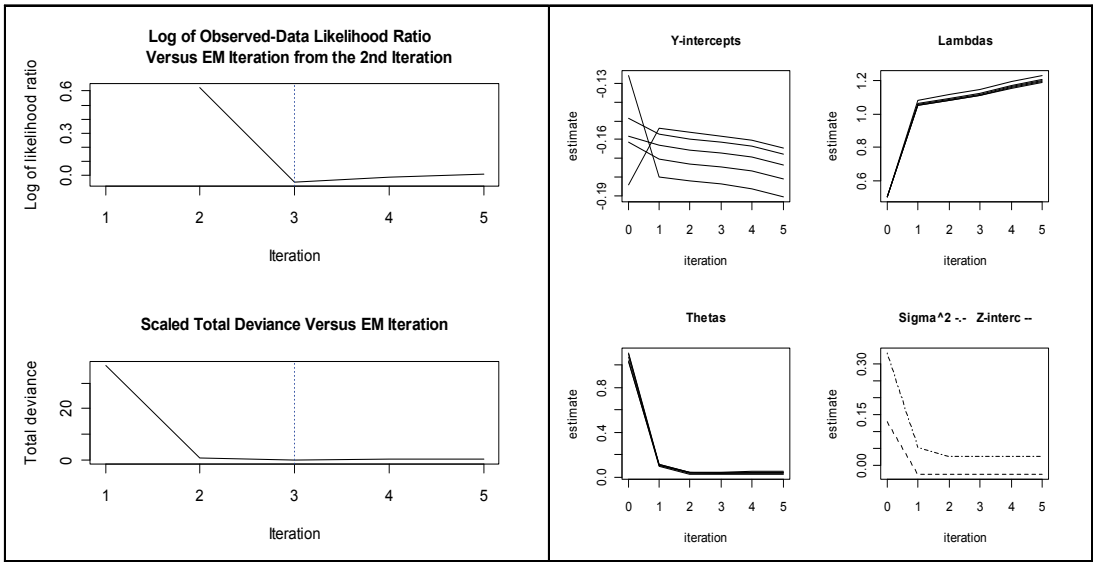


The same analysis (based on a single dataset) is also used to further illustrate with an example of how convergence is decided in a simple run of the ALV algorithm with the results of the monitoring displayed in Figure 5.9. Based on our criteria for convergence, the total deviance

trajectory reached the minimum at the 3rd iteration (see left panels, bottom plot). Therefore convergence was decided after just three iterations and the algorithm was terminated after five iterations (two consecutive iterations following the minimum deviance point). It is seen that the approximate log of likelihood ratio has already reached the region of zero at the chosen convergence point. Similarly the right column shows that the estimates of the residual variances and z-intercept stabilized by the 3rd iteration. However the measurement y-intercepts and slopes (lambdas) continue to drift slightly in their estimates, as previously noted. Note that the recorded values of the log of likelihood ratio start from 2nd iteration (first likelihood ratio being between the first two iterations). For the plots in the right panels the recorded values at zero iteration correspond to the start values used in the ALV algorithm.

Figure 5.9 ALV Model Convergence (Scheme 1, N=300)

Conditional on η, Y's are Linearly Related to Z

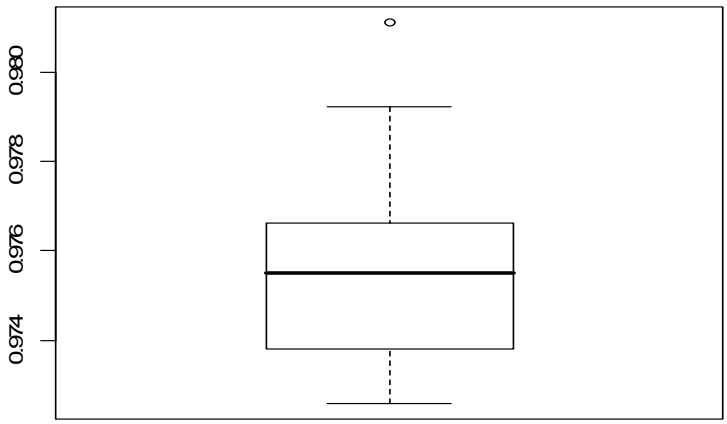


5.3.2 Performance under Scheme 2

As previously stated in section 5.1, the emphasis in the evaluation of ALV performance under scheme 2 is on how well the ALV model is able to estimate η conditioned on Y's and Z and uncover an unknown complex relationship between η and Z. Although we were able simulate a complex relationship accordingly we did not have a full analytic expression for the conditional distribution of η or an appropriate existing standard statistical model for comparison (like in the linear case in scheme 1). Therefore for this assessment we compared the ALV model estimate $\hat{\eta}$ to the true η generated according to scheme 2 specifications based on 50 replications; considering that the marginal distribution of true η is directly proportional to its true conditional distribution to be estimated as $\hat{\eta}$ by the ALV model. We computed 50 correlation values between η and $\hat{\eta}_r$, $r=1,...,50$ using a sample size N = 300. We believe that the strength of the computed correlation indirectly reflects the closeness in values of $\hat{\eta}$ (estimated conditional distribution) to the unknown true values of the conditional distribution. Our results show that the correlation between η and $\hat{\eta}$ is very high in the range of .973 to .981 with mean of .975. The distribution of the calculated correlations is shown in Figure 5.10. This result indicates that the measurement component of the ALV model consistently recovers the latent factor $\hat{\eta}$ underlying the Y's and Z variables even when the solutions to the measurement parameters (y-intercepts and y-slopes) may not be unique. However we are aware that while a high correlation between η and $\hat{\eta}$ is desirable it does not necessarily indicate accuracy in the estimation of $\hat{\eta}$ because a shift of $\hat{\eta}$ from its true value by a constant (bias) can retain the high correlation. Therefore we took the next step to address this concern.

Figure 5.10 Correlations between ALV Estimates of Eta and Population Values

(50 Replications; Scheme 2: Conditional on η, Y's are Nonlinearly Related to Z (Logit); N = 300)



To further quantify the performance of ALV model in estimating $\hat{\eta}$ we considered computing the mean square error or MSE which assesses the quality of the estimation in terms of its variation and unbiasedness. Ideally we would define $MSE(\hat{\eta}) = E[(\hat{\eta} - \eta)^2]$ however this definition of MSE is not appropriate here because $\hat{\eta}$ is not an estimate of the marginal distribution of η, rather it is an estimate of the conditional distribution. Therefore instead we compared the MSE we'd get if we used the true η in a regression model (GAM), to the MSE obtained by using (1) ALV estimated $\hat{\eta}$ and (2) $\eta^* = \eta +$ measurement error where the error term is defined as the ratio of the variance of $\hat{\eta}$ to the variance of true η. For this comparison three different GAM's were fitted to 50 replicated datasets (N=300) generated under scheme 2. The regression models were specified (with variables represented as vectors) as follows:

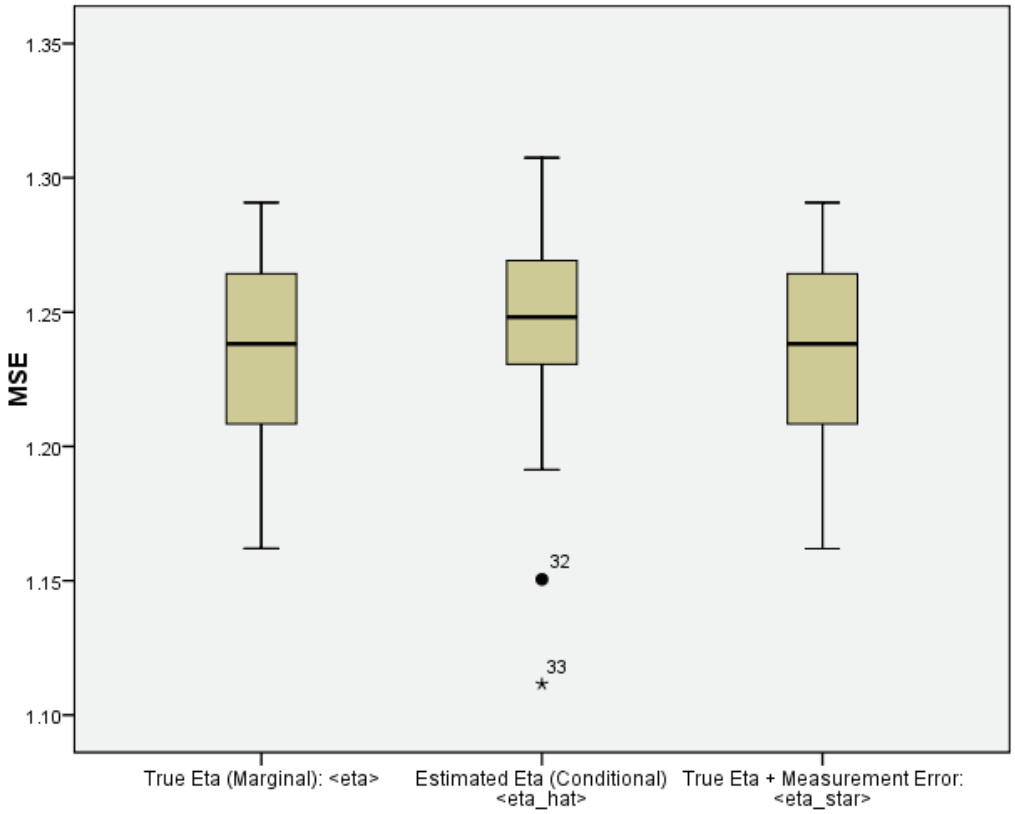
$$
\begin{aligned}
&\text{logit}(Z) \sim s(\eta) + \beta\, GRP + s(\eta \times GRP) + e; \\
&\text{logit}(Z_r) \sim s(\hat{\eta}_r) + \beta\, GRP_r + s(\hat{\eta}_r \times GRP_r) + e_r; \quad \hat{\eta}_r = (\eta_r \mid y_r, z_r; \Omega); \\
&\text{logit}(Z_r) \sim s(\eta_r^*) + \beta\, GRP_r + s(\eta_r^* \times GRP_r) + e_i; \quad \eta_r^* = \eta_r + [\text{var}(\hat{\eta}_r) / \text{var}(\eta_r)]; \\
&r = 1, ..., 50 \text{ replications.}
\end{aligned}
\tag{5.3}
$$

For each fitted GAM the MSE was computed as the mean of the squared residuals; residuals being the difference between the observed and the fitted values of Z. The degree of closeness of the computed MSE's for the different GAMs will reflect the accuracy of the ALV model; that is one can assess how comparable is the estimated $\hat{\eta}$ to the true η in predicting the Z observations. Similar comparison between η and η^* will allow us to assess the effects of measurement errors (associated with $\hat{\eta}$) on the quality of prediction of the true η. Since the MSE in the context of statistical models depends on data, it is treated as a random variable and the 50 replicated MSEs then serve as a measure of how well the three models explain the variability in the observations.

As anticipated, the boxplots of MSEs in Figure 5.11 and the related summaries in Table 5.4 show that generally there is only a slight increase in the MSEs with respect to the predictor $\hat{\eta}$ over that of η, and on the average the increase in median MSE is less than 1%. Also, the measurement errors arising from the ALV estimation did not affect the quality of model prediction when added to the true η in the GAM. In addition the spread of MSE's for $\hat{\eta}$ is slightly smaller than the other two.

Figure 5.11 Boxplots for the MSE's of GAM of Z Separately on η, $\hat{\eta}$ and η^*

(Scheme 2; N=300, 50 Replications)



Having assessed the performance of the ALV model quantitatively, next we wish to use graphical tools to visually demonstrate the primary purpose of the ALV model, which is to assess variation in intervention impact across the unobserved baseline η given an unknown complex relationship between the outcome Z and the predictors including baseline-treatment interaction (η and G). In the following description we again specifically investigated the GAM component of

the ALV to determine how well it is able to recover the 'true' complex relationships between a binary response Z and the predictors. To achieve this we compared two analyses.

Table 5.4 Percent Change in MSEs: GAMs of Z on $\hat{\eta}$ and η^* Compared to η

(Scheme 2; N=300; 50 Replications)

	η	$\hat{\eta}$		η^*	
		Value	Change (%)	Value	Change (%)
Lower Quartile	1.208	1.231	1.836	1.208	-0.001
Median	1.238	1.248	0.806	1.238	0.001
Upper Quartile	1.264	1.269	0.388	1.264	0.000

η= True (simulated); $\hat{\eta}$= ALV estimate $(\widehat{\eta \mid y,z})$; $\eta^* = \eta + \text{var}(\hat{\eta}) / \text{var}(\eta)$

In the first analysis a stand-alone GAM procedure was performed on the sample data using the true η as known. For the second analysis the ALV model was fitted to the same data with $\hat{\eta}$ estimated from the data. This pair of analyses was performed on a single sample randomly selected from 50 under each sample size N = 100, 200, 300; and the results are graphically displayed in Figures 5.12 and 5.13. In the first column of Figure 5.12 the outcome is model estimated logit of Z (simulated as binary) and is plotted against the true η that was used to generate it. This plot is used to establish the true trajectories according to the simulation model in scheme 2. The ALV model performance is evaluated against the true trajectories directly by comparing plots in the first and third columns. Also, the trajectories of the fitted values by ALV model (column 3) are compared to those of the stand alone GAM (column 2). Each trajectory on

a single plot represents members of one arm of treatment. The curves are drawn with points corresponding to actual data points (η and estimated logit of Z). Vertical dashed lines are drawn to partition the trajectories along the indicated quantiles of η. The vertical lines serve as aids in the assessment of distribution of fitted values for individual subjects across the baseline; also comparison across modalities is made easy. Confidence bounds are constructed at one standard error around the estimates for easy comparison on precision of estimates.

From the patterns of the plots (Figure 5.12), compared to the true trajectories both GAM and ALV trajectories reveal some attenuation generally; otherwise the ALV trajectories are nearly identical to those of GAM. Note that there are one or two substantial outliers in the observed Z (logit transformed) located in the top right corners in column one. The presence of such outliers in data has been noted to be problematic in GAM fitting technique (Wood, 2006), apparently the outliers were not tracked to any reasonable degree by the trajectories produced by both GAM and ALV model. Both methods did not completely capture the true relationship between Z and η, however the use of a single dataset as a basis for the comparison prevents any definitive conclusion here. Possibly these performances may also be related to the outlier problems. Single replication analyses notwithstanding, the similarity between GAM and ALV models reflects our earlier findings (comparisons of MSEs) and suggests that when η and its relationship Z are unknown, the ALV model may perform equivalently to GAM procedure given known η. Also graphically there seems an improved performance by both GAM and ALV models (closer approximation to the true trajectories) and increased similarities between the two as sample size increases (Figure 5.12).

The convergence pattern of the ALV analysis under scheme 2 is again depicted in Figure 5.13. For example, the ALV model converged after 4 iterations (left column). On the right

column it is seen that the parameter estimates (except for the lambdas) have stabilized before the convergence point. These findings are similar to those obtained under scheme 1.

Figure 5.12 Plots of Z (Binary Outcome) Predicted by Eta (Baseline Risk) by G (Treatment) (Results Based on a Single Simulated Sample Among 50)

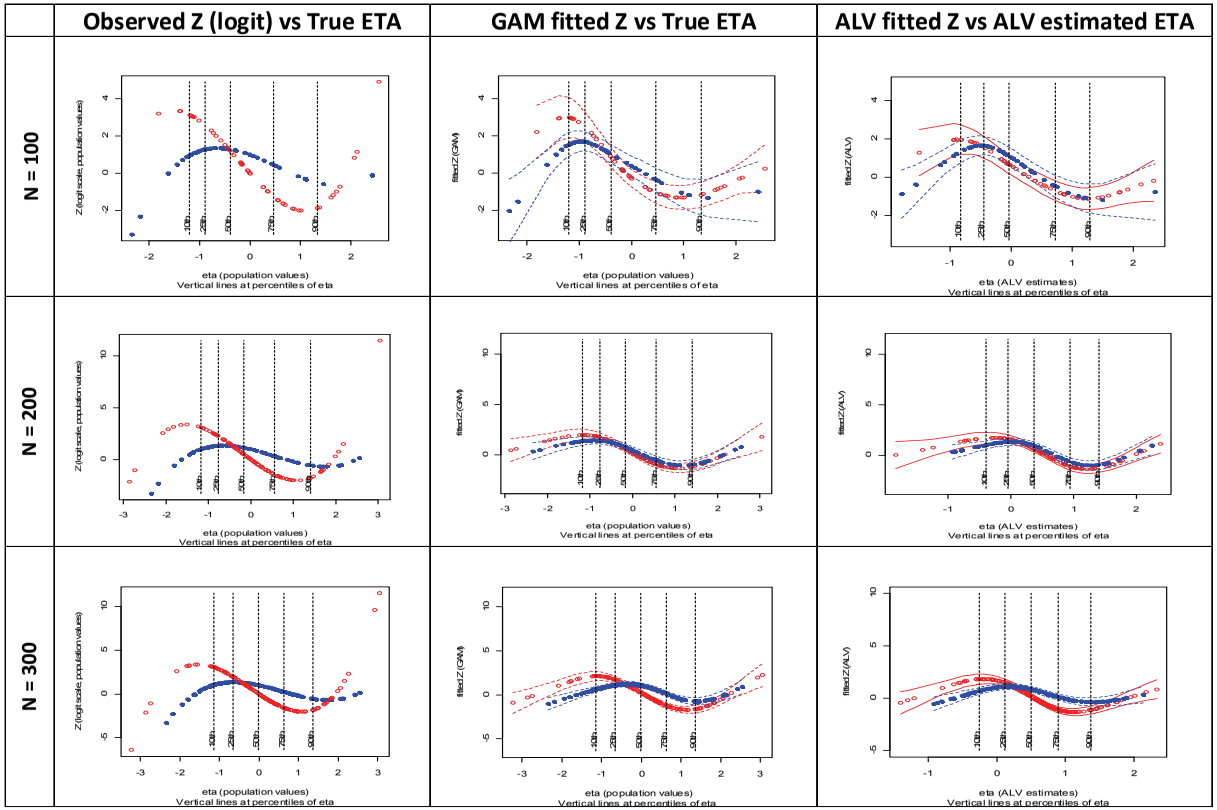
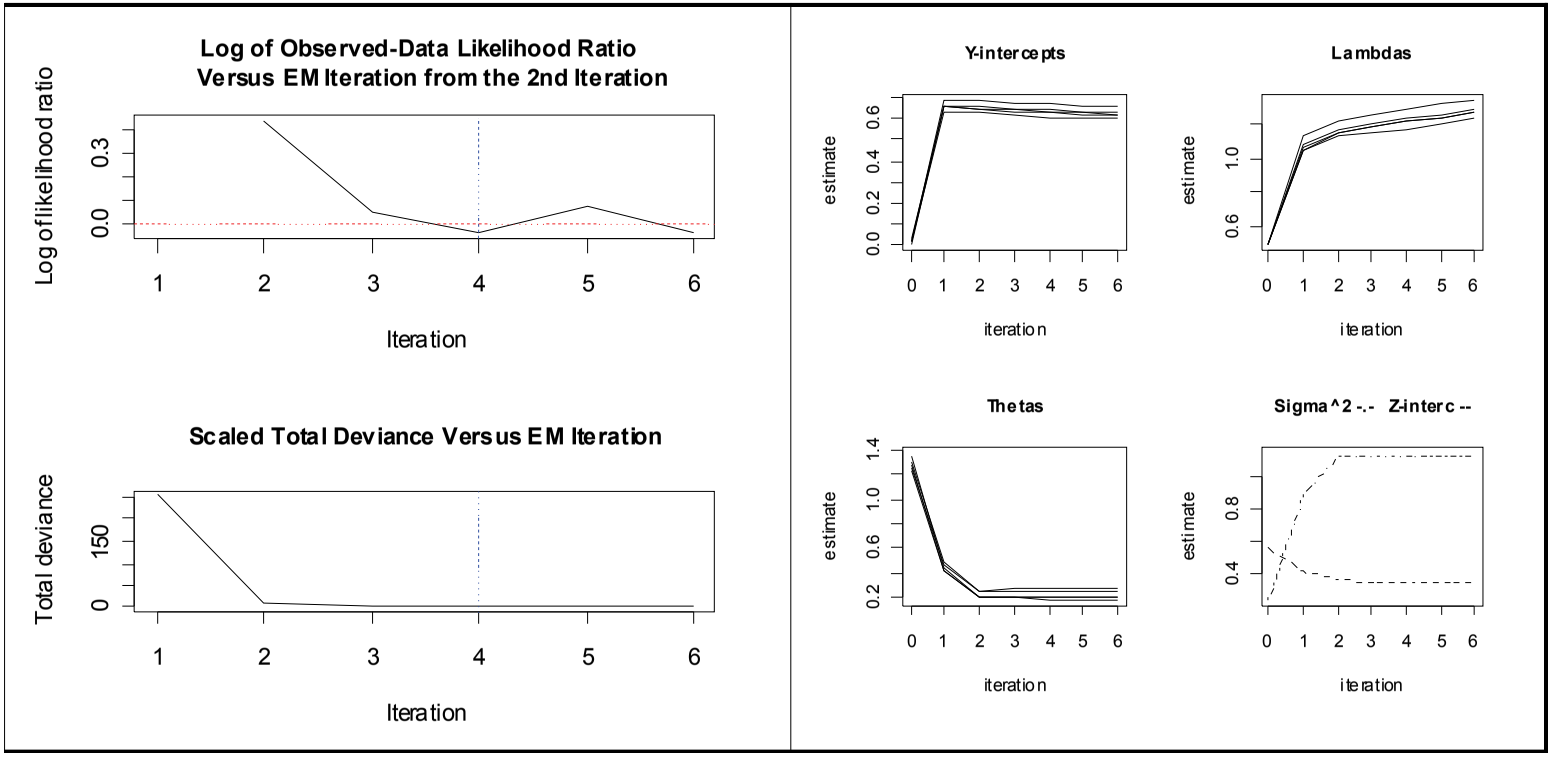


Figure 5.13. ALV Model Convergence (Results Based on a Single Simulated Sample of Size N=200 Selected from 50)



CHAPTER 6

APPLICATION OF ALV MODEL TO ASAPS DATA

We illustrate the ALV method with an application to data from the Adolescent Substance Abuse Prevention Study (ASAPS) (Sloboda, et al., 2008). This is a cluster randomized field study involving 19,200 students in 83 high school clusters (a cluster being a high school and all its feeder middle schools) from six metropolitan areas across the U.S. (see chapter 1of this dissertation). The study's main objective was to test an intervention program Take Charge of Your Life (TCYL) delivered by selected trained D.A.R.E. officers, on its effectiveness in reducing some key behavioral outcomes: use of alcohol, tobacco and other drugs (ATOD). One of the major research questions was to investigate who benefits or is harmed by the instituted intervention program and how the intervention effects are moderated by the baseline risk factors. The original D.A.R.E. curriculum was criticized for focusing on the low risk group, thinking that high risk group would be alienated by officers who were "preaching at them". The new curriculum with TCYL program delivered by trained instructors was designed with sensation seeking and high risk kids in mind. The aim was to impact intentions to use alcohol, tobacco and other drugs (marijuana) by addressing baseline (7^{th} grade pretest) beliefs as to the normative use of ATOD; perceptions of the harmful effects of use; and skills necessary to avoid substance use (decision making, resistance skills). It was hypothesized that intervention may show different effects for low and high risk kids at baseline.

The 1^{st} wave-data (pretest data) consisted of 53 items that showed significant loadings on 10 risk constructs in a previous factor analysis performed by the researchers. The item-level response scores on Likert scale were coded so that the highest score implies highest risk. As an example of the constructs, the five items designed to assess normative beliefs of 11^{th} graders about alcohol,

tobacco and other drugs are displayed in Table 6.1. To illustrate ALV model application these constructs (see Table 6.2) formed 10 summary risk variables that served as factor indicators for a single latent risk to be estimated by ALV. Some of these summary risk variables are skewed but no attempt was made to dichotomize any of them since the ALV model its current form only takes continuous measurement variables. In the ALV analysis we examined variation in the intervention program effect on only one of the 7th wave-outcomes (substance use in 11th grade), (see Table 6.2), across the estimated baseline risk. This illustrative ALV analysis is neither complete nor final because of the presence of significant amount of missing data on the outcomes, for which no imputation was performed (Table 6.3). The researchers had anticipated 50 percent attrition among the student cohort. There was substantial cross mobility of students during transition to high school from feeder middle schools. For example some students went into study high schools not assigned to their middle schools or to high schools not included in the study. In addition, one high school opted out of the study and by the time of the 11th grade survey two additional high schools affected by Hurricane Katrina were lost from the study. Therefore, for illustrative purpose, we report here on the results of fitting ALV model to incomplete data on risk measures in 7th grade and substance use in 11th grade for 2500 males from the ASAPS study (after listwise deletion of missing values).

Table 6.1 Five Items Used in the ASAPS to Assess Normative Beliefs of 11th Graders

Item Questions	
In the Last 30 Days, how many 8th graders across the entire U.S. do you think …. a) used cocaine or other hard drugs? b) drank beer, wine or liquor? c) smoked cigarettes? d) sniffed glue, inhale gases or a spray to get high? e) smoked marijuana (pot, reefer, weed, blunts)?	
Possible Answers	Possible Scores
All or almost all (100%)	5
More than half (about 75%)	4
About half (50%)	3
Less than half (25%)	2
None (0%)	1

Table 6.2 Ten Summary Baseline Risk Constructs in ASAPS Data

	Construct
1	Normative beliefs
2	Referent others
3	Consequences of ATOD use on the brain
4	Personal attitudes towards ATOD use
5	Negative expectation from ATOD use
6	Intentions (to use under certain situations)
7	Intentions (what age ok to initiate risky behave)
8	Number of best friends using ATOD
9	Pro-social bonding (school attachment)
10	Self-reported delinquent behaviors

ATOD = alcohol, tobacco and other drugs (marijuana)

Table 6.3 Some 11th Grade Outcomes and Missing Data in ASAPS Data

	# Missing	Proportion Missing
Explanatory Variables		
School	0	0
Gender	0	0
Treatment	476	0.03
Outcomes		
Used Marijuana in Past 30 Days	7869	0.46
Got Drunk in Past 30 Days	7824	0.46
Binge drinking in Past 30 Days	7758	0.46
Used Cigs in Past 30 Days	7750	0.45
Used Inhalants in Past 30 Days	7826	0.46

A key feature of the ALV model lending weight to its appropriateness for analyzing the ASAPS data is that it can easily handle complex relationships in the data without requiring the knowledge of the relationship beforehand. Nonlinearities arise in the data because of the potential variation in impact of the administered behavioral intervention on the individuals with different baseline risk experience. It is also important to note that the risk experience was not directly observed and has to be inferred from the data as a latent variable; plus, the shape of the relationship between the latent risk and the outcome (in this example, marijuana use) is unknown and is potentially complex. These are compelling reasons to specify the effects of the latent baseline risk (and its interaction with intervention) nonparametrically. To include the cluster effects of school districts in the analysis Generalized Additive Mixed Model (GAMM) was specified for the additive part of the ALV algorithm at the final EM iteration after baseline risk has been estimated ($\hat{\eta}$) from the data, treating the clusters as random effects:

$$\begin{aligned} \text{logit}(E[z_i]) &= \beta_0 + \beta_1 \text{Group}_i + s(\widehat{\text{Risk}}_i) + s(\widehat{\text{Risk}}_i * \text{Group}_i) + H_i b + \varepsilon_i; \\ b &\sim N(0, \sigma_b); \varepsilon_i \sim N(0, \sigma_\varepsilon). \end{aligned} \tag{5.4}$$

Here z is a binary response 'Marijuana Use'; β's are fixed parameters for the model intercept and intervention group variable; *s(.)* is a smoothing function that estimates the unknown complex relationships of the response to the baseline risk and its interaction with treatment; *H* and *b* are the random effects model matrix and coefficients.

The partial results (additive part) of ALV model fit to the ASAPS data are reported here (Figures 6.1 & 6.2; Tables 6.4 a & b). In the context of the estimates of the nonparametric functions, the plots in Figure 6.1 describe the relationships between the smoothing terms in the model and the outcome using solid lines/curves within 95% point wise confidence bands (dashed lines). Along the bottom of each plot are rug-plots at points corresponding to the covariate values for each smooth. For the whole sample (treatment and control), a smooth curve (top panel) is estimated with 2.97 (number in y-axis caption) effective degrees of freedom for the effect of baseline risk while the estimated interaction effect (bottom panel) is approximately linear with the outcome and so requires only 1 degree of freedom to estimate a slope. The above information could be missed if a parametric model with *s(.)* restricted to be linear were to be fitted to the data; although for this particular sample data, the fit of a quadratic model may be sufficiently close in quality to the ALV model fit.

Figure 6.1 Estimated Relationship of Probability of Marijuana Use to Baseline Risk.

ALV Estimated Baseline Risk (top panel) and Baseline-Treatment Interaction (bottom panel)

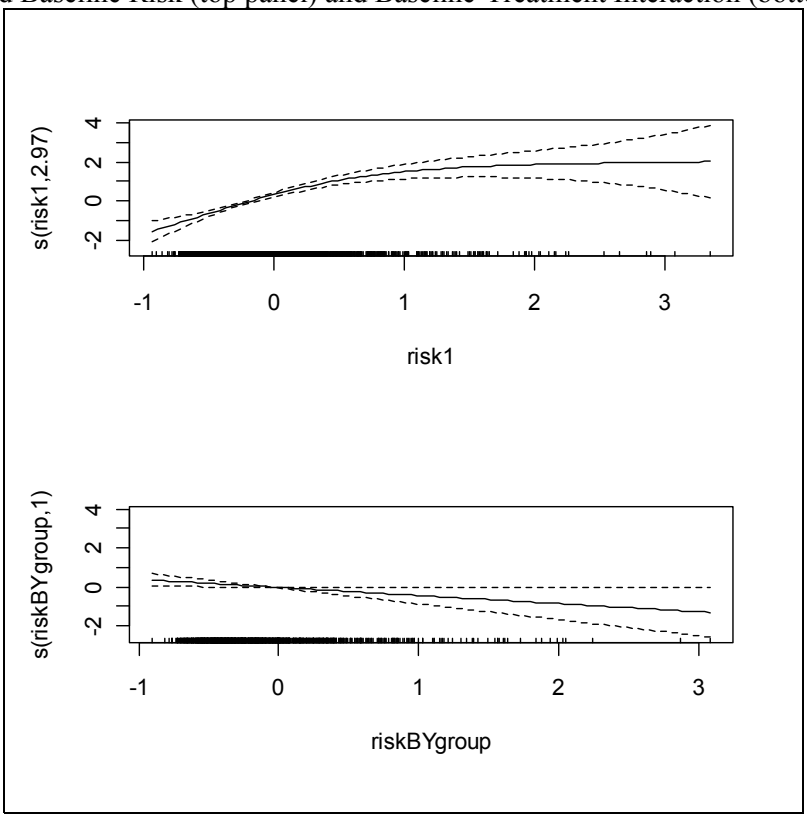
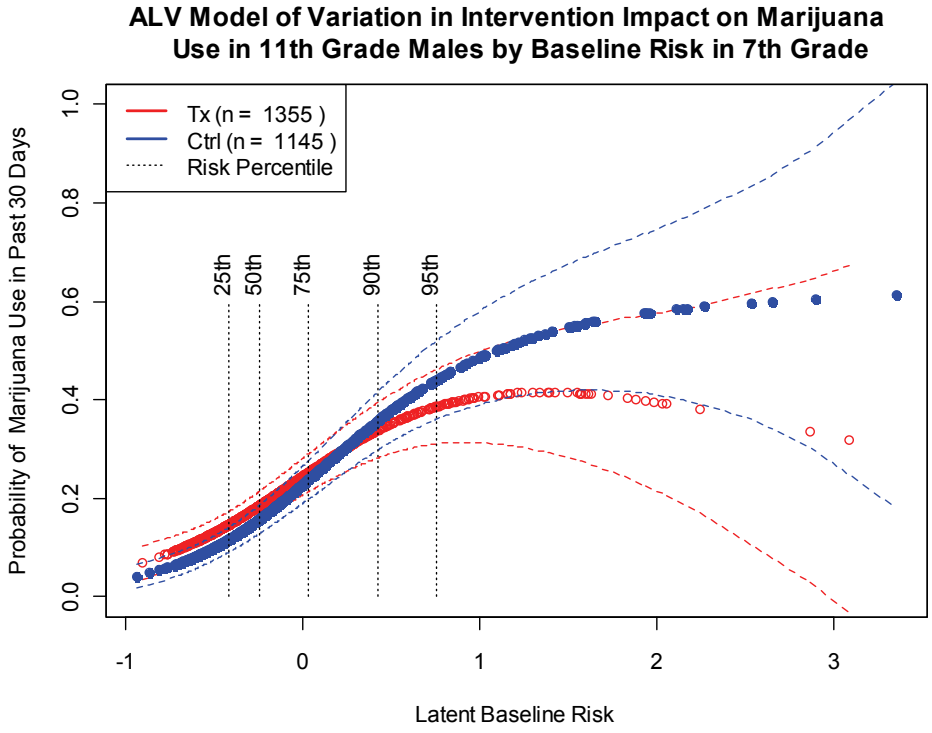


Figure 6.2 is a visual display of the variation in intervention impact across the baseline risk. The dashed curves represent pointwise 95% confidence intervals around values predicted from the results of the fitted GAMM. To compute these values the R function *predict()* was applied to the R object for GAMM fit; the corresponding standard errors were also returned. The 95% confidence was then constructed around each predicted value as *value +/- standard error* and from these generated values it was possible to draw the upper and lower limits separately around the fitted curves. The plot shows a changing direction of intervention effects along the risk scale and precisely which levels of risk are associated with higher or lower marijuana use. Note

that the uniformly increasing group difference in average probability of marijuana use across baseline indicates a linear interaction effect, as revealed by the bottom panel plot in Figure 6.1, and previously expounded in this dissertation (refer to Figure 3.1 (B)). For the top 5% of kids on the baseline risk scale, the average probability of marijuana use is obviously lower for individuals in the intervention group relative to the controls. In contrast, the intervention appears to be marginally harmful to the low risk subgroup (below 25 percentile). In summary the effect of the intervention is harmful when there is low baseline risk and gets more beneficial with higher risk. However only across the percentiles where the 95% confidence intervals show no overlap is significant intervention impact implied. There appears to be some degree of overlap across all percentiles more marked at the top end of the risk scale. This indicates that the intervention effects are not locally significant, that is the intervention has no significant impact on any risk subgroup at the 95% confidence level.

Figure 6.2 The Effect of Baseline Risk on Probability of Marijuana Use by Group, with 95% Pointwise Confidence Bands



Results of the ALV model analysis are also reported in Tables 6.4 a & b. These results are from the direct output of the Additive component of the ALV model and are supported by the iterpretations derived from Figure 6.1. In Table 6.4a both terms for the baseline risk and the interactioe are specified as nonparametric smoothing functions as in (5.4). Under the section on Nonlinear Terms the baseline risk (3rd row) shows significant nonlinearity ($p<0.001$) in its relationship with marijuana use and this effect is estimated as a smooth curve with 2.97 expected degrees of freedom (edf). However a straight line corresponding to edf of 1.00 is estimated for its interaction effect (4th row) and the test of nonlinearity for this term is not significant ($p=0.07$). It should be noted here that the p-values of smooth terms are only approximate due to the

uncertainty in estimating smoothing parameters (Wood, 2006). According to the author, the p-values are usually safe to rely on only when they give a very clear cut result; when the results are around a reject/accept threshold, the tests reject the null too readily and therefore must be treated with caution. Given that linear interaction effect is demonstrated in Table 6.4a, we fitted another GAMM this time using a fixed parameter for the interaction term; the results (Table 6.4b, 3rd row) show a negative linear interaction that is fairly significant (p=0.037) at the 95% confidence level. This is in support of the finding of reversal of intervention effects along the baseline demonstrated graphically in Figure 6.2; and given the caution required for interpreting p-values, the interaction effects are probably not significant.

For the parametric terms, we see in the 2nd rows of both tables that no significant main effect (p=0.50) is demonstrated for the intervention. Finally, there is significant random effect of school districts clustering in the data (last rows). Combining all of the findings from Figures 6.1 & 6.2 and Tables 6.4a&b, in summary there is significant nonlinearity in the relationship of 7th grade baseline risk and Marijuana use in 11th grade but no significant intervention effect are demonstrated across any baseline risk subgroups.

Table 6.4a ALV Model of Marijuana Use Reported by 11th Grade Males (N=2500; 79 High School Clusters): Additive Sub-model# Includes Nonlinear Interaction Term

Type of Effect	Effect	Coefficient (Logit)	SE	*z*-value	*p*-value
Parametric Terms					
1. Intercept		-1.521	0.100	-15.103	<0.001
2. Intervention Main Effect (adjusted)	Intervention = 1 vs. Controls = 0	0.091	0.135	0.675	0.500
Smooth Terms		**Functions**	***edf***	**F**	***p*-value**
3. Baseline Risk	Smooth (baseline)	Smoothing coefficients	2.97	27.493	<0.001
4. Interaction Effect	Smooth (interaction)	Smoothing coefficients	1.00	2.906	0.070
Random Effects	**Effect Name**	**SD**	**95% CI**		
Cluster	School District	0.348	0.219 – 0.555		

#GAMM: $\log it(E[\text{Marijuana Use}_i]) = \beta_0 + \beta_1 * \text{Intervention}_i + s(\text{Risk}_i) + s(\text{Risk}_i * \text{Intervention}_i)$

Table 6.4b ALV Model of Marijuana Use Reported by 11th Grade Males (N=2500; 79 High School Clusters): Additive Sub-model# Includes Linear Interaction Term

Type of Effect	Effect	Coefficient (Logit)	SE	*z*-value	*p*-value
Parametric Terms					
1. Intercept		-1.552	0.102	-15.168	<0.001
2. Intervention Main Effect (adjusted)	Intervention = 1 vs. Controls = 0	0.091	0.135	0.675	0.500
3. Interaction Effects	Interv-by-Baseline	-0.418	0.200	-2.087	0.037
Smooth Terms		**Functions**	***edf***	**F**	***p*-value**
4. Baseline Risk	Smooth (baseline)	Smoothing coefficients	2.97	27.480	<0.001
Random Effects	**Effect Name**	**SD**	**95% CI**		
Cluster	School District	0.348	0.219 – 0.555		

#GAMM: $\log it(E[\text{Marijuana Use}_i]) = \beta_0 + \beta_1 * \text{Intervention}_i + s(\text{Risk}_i) + \beta_3(\text{Risk}_i * \text{Intervention}_i)$

CHAPTER 7

DISSCUSSION & RECOMMENDATIONS

In this dissertation we have considered plausible variations in intervention impact due to baseline individual level risk/protective factor characteristics. We also considered the importance of modeling these variations in the statistical analyses of behavioral, social and psychological research data from randomized field trials in particular, where measurement errors and nonlinearity commonly arise and pose statistical challenges. We reviewed the existing statistical modeling techniques that have been applied to assess these variations, such as nonlinear (polynomial terms) SEM and GAM. We highlighted their limitations including the inefficiency associated with the ad hoc approach of stepwise application of these two methods in one analysis but on different statistical application platforms. To address these challenges we have developed a new modeling technique, ALV, by integrating the two powerful statistical models (SEM and GAM) into one model that runs on one platform and draws strength from both methods.

We reached the following conclusions from the results of our simulation studies. First, the ALV model works well with the tested sample sizes of 100, 200, and 300 with measurement errors. Second, this new method was successful in capturing the nonlinear dependence of the outcome on a latent variable in the data. Also the method performs nonlinear modeling task nearly as well as it does a linear modeling at least in the simulation studies with sample size as low as 100.

Like most existing methods in SEM our proposed ALV model approach is based on the assumptions of conditional independence for the baseline factor indicators and distal outcome given the underlying latent factor, plus normally distributed errors. However a notable

distinguishing feature of the ALV modeling technique is that it makes no assumption about the relationship between the latent factor and the distal outcome. The new ALV method is developed to simultaneously estimate the latent factor underlying the observed baseline risk variables plus the complex relationship between the latent factor and the distal outcome it predicts, without requiring a priori specification of a functional form for the unknown relationship. The ALV modeling is implemented in Monte Carlo EM environment and it involves the estimation of posterior distribution of the latent factor in the E-step via Metropolis algorithm while ML estimation of parameters is via standard regression sub-models in the M-step. The EM type algorithms are tremendously useful in solving statistical problems involving missing and latent data.

In order to establish a more objective criterion for our stopping rule for convergence in the Monte Carlo EM loop within the ALV algorithm, we have taken into account the overall fit of the ALV model in addition to the behavior of parameters. Given the typical long term pattern of the ALV model deviance trace with respect to EM iterations, we are able to conclude model convergence at the point of minimum deviance, which we consider to be probably global within the context of our simulations. Our stopping rule is new relative to those proposed in the literature for Monte Carlo EM; and from our experience we also found our criteria (including point of minimum deviance) to be very crucial for the efficiency of the ALV algorithm. The criteria allow us to decide convergence after single digit number of EM iterations in most instances, because the ALM model is largely a linear model.

Performance-wise, a key emphasis has been on testing the ability of ALV model to accurately recover both the latent factor (underlying baseline risk) as well as the complex nonlinear relationships between the outcome and the predictors. The results of our simulation studies show that the ALV model performs well. While the role of the measurement part is

mainly concerned with estimation of the latent factor, for interpretability our focus necessarily shifts to the nonparametric (GAM) component, on which the major feature of the ALV model depends. Compared to the easily interpretable GLMs, GAMs may be more difficult to interpret because of the nonparametric nature of the underlying nonlinearity in the data. However it is important to acknowledge that although GAM's may serve different analytic purposes like suitably exploring the data nonparametrically and visualizing the complex relationships, in the presence of unknown complex nonlinearity GAM's are closer to reality and are known to yield a better fit than their GLM counterparts. These properties are well illuminated by the results of our application of the proposed ALV model to both simulated and real data in this dissertation. In practice, because of the flexibility of GAM technique, it is very possible to provide a good fit to the data by tracking significant noise in addition to the nonlinear relationships in the predictor variables. This happens whenever higher than the appropriate degrees of freedom are used in estimating the nonparametric functions of the predictor terms. Although the user is allowed to specify degrees of freedom for the cubic spline smoother for each predictor term in a stand-alone GAM procedure, the optional feature we adopted in the GAM component of ALV model allows for optimal estimates of effective degrees of freedom to be computed directly by the model (Wood, 2006). So the potential problems of over fitting (or under fitting) typically associated with user-specified degrees of freedom in AM methods are minimized in the ALV method.

One major limitation was the number of cases we examined in the simulation. This limitation with respect to maximum size of 300 was due to practical considerations since each simulation required massive computing time. The minimum size of 100 was chosen because typically factor analysis is a large sample procedure, and also because the choice is in line with similar past studies involving Monte Carlo version of the EM (Lee & Song, 2007; Lee & Zhu, 2002). However more studies are required to study the stability of ALV model when sample size

drops below the minimum of 100 used in the present study. Another major limitation of the ALV model in its current form is its listwise deletion approach to missing data problems. Given the frequent encounter with missing data in practice and the availability of more effective methods of handling this problem, the incorporation of such methods into ALV model will be of considerable importance and we are planning to do this in our next stage. As it is currently set up, the nesting in the data is accounted for only at the final EM iteration and only in the GAM component of the ALV model. Further studies are needed to assess the adequacy of this partial effort compared to full multilevel extensions to the ALV model. Although this new approach is computationally intensive, given the persistent rapid developments in computer technology, this should not be considered a serious limitation. Even though the ALV model consistently estimates the latent factor accurately in the measurement part of the model, the associated measurement parameter estimates are not stable and this may indicate that the solutions are non-unique. Therefore the emphasis of the ALV model application should be on the accurate recovery of unknown complex relationships in the data; in its current form it may not be useful for analyzing psychometric properties of instruments.

There are several other ways (than our choice in this dissertation) of defining a cubic regression spline basis which may offer some advantages with respect to the interpretability of the parameters and appropriateness to the data at hand (Wood, 2006). The ALV method can be improved upon therefore by exploring other smoothing spline bases available as options in the R package *mgcv* and determining under what conditions a particular choice would be best within the ALV framework.

Our model can be extended to examine complex nonlinearity between multiple distal outcomes and their predictors including multiple latent factors (e.g. multiple-factors solutions to observed baseline risk variables) or growth factors in a longitudinal study. In future we intend to

also explore the application of ALV method to a wider spectrum of nonlinear structural equation modeling involving complex factor-to-factor, factor-to-indicator, and indicator-to-indicator relationships, using nonparametric methods.

In conclusion, the ALV modeling technique allows researchers to assess how an intervention affects individuals differently as a function of baseline risk that is itself measured with error, and uncover complex relationships in the data that might otherwise be missed. In practice, its users are relieved from the need to decide functional forms for the complex relationships before the model is run. The ALV program is written in R language and the R software is freely available; so general users can apply the new methodology. We expect the ALV model and its extensions to have lots of new applications to modeling of behavioral, sociological and psychological data in the future.

REFERENCES CITED

Brown, C. (1993). Analyzing Preventive Trials with Generalized Additive Models. *American Journal of Community Psychology , 21*, 635-664.

Brown, C., Wang, W., Kellam, S., Petras, H., Toyinbo, P., Poduska, J., et al. (2008). Methods for Testing Theory and Evaluating Impact in Randomized Field Trials: Intent-to-Treat Analyses for Integrating the Perspectives of Person, Place, and Time. *Drug and Alcohol Dependence , 95* (Supplement 1), S74-S104.

Chambers, J. M., & Hastie, T. J. (1993). *Statistical Models in S.* London: Chapman & Hall.

Cheney, W., & Kincaid, D. (2004). *Numerical Mathematics and Computing.* Belmont, California: Brooks/Cole-Thomson.

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician , 49* (4), 327-335.

Chib, S., & Jeliazkov, I. (2006). Inference in Semiparametric Dynamic Models for Binary Longitudinal Data. *Journal of the American Statistical Association , 101*, 685-700.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society , 39* (1), 1-38.

Gamerman, D., & Lopes, H. F. (2006). *Markov Chain Monte Carlo. Stochastic Simulation for Bayesian Inference* (2nd Edition ed.). Boca Raton, FL: Chapman & Hall/CRC.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEE Transactions on Pattern Analysis and Machine Intelligence , 6*, 721–741.

Gu, C. (2002). *Smoothing Spline Anova Models.* New York: Springer-Verlag.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized Additive Models.* New York: Chapman and Hall.

Heath, M. T. (2005). *Scientific Computing : An Introductory Survey.* New York, New York: McGraw-Hill.

Holler, K. D. (2005). *Generalized Additive Models*. Retrieved May 5, 2009, from www.casact.org/education/specsem/f2005/handouts/holler.ppt

Joreskog, K. G. (1977). Structural Equation Models in the Social Sciences: Specification, Estimation and Testing. In P. R. Krishnaiah, *Applications of Statistics.* Armsterdam: North-Holland.

Kellam, S., Brown, C., Poduska, J., Ialongo, N., Petras, H., Wang, W., et al. (2008). Effects of a Universal Classroom Behavior Management Program in First and Second Grades on Young Adult Behavioral, Psychiatric, and Social Outcomes. *Drug and Alcohol Dependence , 95* (Supplement 1), S5–S28.

Khoo, S.-T. (1997). Assessing Interactions between Program Effects and Baseline. *Dissertation* . University of California, Los Angeles, Carlifornia.

Law, N. J., Taylor, J. M., & Sandler, H. (2002). The Joint Modeling of a Longitudinal Disease Progression Marker and the Failure Time Process in the Presence of Cure. *Biostatistics , 3* (4), 547-563.

Lee, S. Y., & Zhu, H. T. (2000). Statistical Analysis of Nonlinear Structural Equation Modelwith Continous and Polytomous Data. *British Journal of Mathematical and Statistical Psychology , 53*, 209-232.

Lee, S.-Y., & Song, X.-Y. (2007). A Unified Maximum Likelihood Approach for Analyzing Structural Equation Models With Missing Nonstandard Data. *Sociological Methods & Research , 35* (3), 352-381.

Lee, S.-Y., & Zhu, H.-T. (2002). Maximum Likelihood Estimation of Nonlinear Structural Equation Models. *Psychometrika , 67* (2), 189-210.

Lee, S.-Y., Song, X.-Y., & Lee, J. C. (2003). Maximum Likelihood Estimation of Nonlinear Structural Equation Models with Ignorable Missing Data. *Journal of Educational and Behavioral Statistics , 28* (2), 111-134.

Martin, A. D., Quinn, K. M., & Park, J. H. (2009). *R package version 0.9-6.* Retrieved from MCMCpack: Markov chain Monte Carlo (MCMC) Package: http://mcmcpack.wustl.edu

McLachlan, G. J., & Krishnan, T. (2008). *The EM Algorithm and Extensions* (2nd Edition ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Meng, X. L., & Schiling, S. (1996). Fitting Full-information Item Factor Models and an Empirical Investigation of BridgeSampling. *Journal of American Statistical Association , 91*, 1254-1267.

Muthen, B. O. (2002). Beyond SEM: General Latent Variable Modeling. *Behaviormetrika , 29* (1), 81-117.

Muthen, B. O. (1989). Latent Variable Modeling in Heterogeneous Populations. *Psychometrika , 54* (4), 557-585.

Muthen, L. K., & Muthen, B. O. (2008). *Mplus User's Guide, Version 5* .

Nelder, J. A., & Wedderburn, R. W. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society , 135*, 370-384.

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006, March). CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News* , 7-11.

Poduska, J., Kellam, S. G., Wang, W., Brown, C. H., Ialongo, N., & Toyinbo, P. (2008). Impact of the Good Behavior Game, a Universal Classroom–Based Behavior Intervention, on Young Adult Service Use for Problems with Emotions, Behavior, or Drugs or Alcohol. *Drug and Alcohol Dependence , 95* (Supplement 1), S29-S44.

R Development Core Team. (2008). *R: A Language and Environment for Statistical Computing.* Vienna: R Foundation for Statistical Computing.

Rubin, D. B. (1991). EM and Beyond. *Psychometrika , 56*, 241-254.

Rubin, D. B. (1976). Inference and Missing Data. *Biometrika , 63*, 581-592.

Sloboda, Z., Pyakuryal, A., Stephens, P. C., Teasdale, B., Forrest, D., Stephens, R. C., et al. (2008). Reports of Prevention Programming Available in Schools. *Prev. Sci.*

Song, X.-Y., & Lee, S.-Y. (2005). Maximum Likelihood Analysis of Nonlinear Structural Equation Models With Dichotomous Variables. *Multivariate Behavioral Research , 40* (2), 151-177.

Walsh, B. (2004). Markov Chain Monte Carlo and Gibbs Sampling. *Lecture Notes for EEB 581 , version 26*.

Wang, C., Brown, C., & Bandeen-Roche, K. (2005). Residual Diagnostics for Growth Mixture Models. *Journal of the American Statistical Association* .

Wei, G. C., & Tanner, M. A. (1990). A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation. *American Statistical Association , 85* (411), 699-704.

Wood, S. N. (2006). *Generalized Additive Models: An Introduction with R.* Boca Raton, Florida: Chapman & Hall / CRC.

Xiang, D. (2004). Fitting Generalized Additive Models with the GAM Procedure. *Paper P256-26* . Cary, NC: SAS® 9.1.2 User's Guide, SAS Institute Inc.

APPENDICES

APPENDIX A: DATA SIMULATION CODES FOR SCHEMES 1 & 2

```
#########################################################
# SIMULATE COPIES OF DATASETS (N=100,200,300):
# 6 VARIABLES CONDITIONALLY INDEPENDENT GIVEN ETA:
# {5 CONTINUOUS Y's + 1 CONTINUOUS OR BINARY Z}
#########################################################


# Define Population parameters
n <- 150 # half sample size
J <- 50  # number of datasets
lambda.1 <- lambda.2 <-lambda.3 <-lambda.4 <-lambda.5 <-1
sigma.eta <- 1 # s.d. for eta
sigma.ey <- .5 # s.d. for error term of y
sigma.ez1 <- .5 # s.d. for error term of z (linear model)


# define f1 , f2
f1 <- function ( x ) { 1 - x - 0.5 * x^2 + 0.3 * x^3 }
f2 <- function ( x ) {  - 3 * x + 0.4 * x^2 + 0.6 * x^3}
# simulate eta, the latent factor
set.seed ( 1235 )
eta <- rnorm ( 2*n, 0, sigma.eta )
```

```
# define G (group)

G <- c ( rep (0, n), rep ( 1, n ) )


# create arrays to store J number of (2n x 8) datasets for

#    a) ContArray: Z is cont and linearly related to eta

#    b) BinArray: Z is binary and nonlinearly related to eta


# DATASETS do not include eta column

ContArray <- BinArray       <- matrix(NA, nrow=(2*n), ncol=(7*J))

dim(ContArray) <- dim(BinArray)   <- c((2*n), 7, J)

dimnames(ContArray) <- dimnames(BinArray)<- list(NULL,
c("Y1","Y2","Y3","Y4","Y5","Z","GRP"),1:J )


# DATASETS include eta column

ContArray2 <- BinArray2    <- matrix(NA, nrow=(2*n), ncol=(8*J))

dim(ContArray2) <- dim(BinArray2)           <- c((2*n), 8, J)

dimnames(ContArray2) <- dimnames(BinArray2)<- list(NULL,
c("Y1","Y2","Y3","Y4","Y5","Z","GRP", "ETA"),1:J )
```

APPENDIX A: (CONTINUED)

```
# SIMULATION

    for (j in 1:J){

# simulate y1 to y5

e1 <- rnorm ( 2*n, 0, sigma.ey )

y1 <- lambda.1 * eta + e1

e2 <- rnorm ( 2*n, 0, sigma.ey )

y2 <- lambda.2 * eta + e2

e3 <- rnorm ( 2*n, 0, sigma.ey )

y3 <- lambda.3 * eta + e3

e4 <- rnorm ( 2*n, 0, sigma.ey )

y4 <- lambda.4 * eta + e4

e5 <- rnorm ( 2*n, 0, sigma.ey )

y5 <- lambda.5 * eta + e5

# define Z (continuous, linear with eta) for group (0,1)

ez1 <- rnorm ( 2*n, 0, sigma.ez1 )

Z <- rep (0, 2*n )

Z [1: n] <- 0.2*( eta [1:n] ) + ez1 [1:n]

Z [(n+1) : (2*n) ] <- 0.7*( eta [(n+1) : (2*n) ] ) + ez1 [(n+1) : (2*n) ]

ContArray[,,j] <- cbind ( y1, y2, y3, y4, y5, Z, G )

ContArray2[,,j] <- cbind ( y1, y2, y3, y4, y5, Z, G, eta )
```

```
#-----------------------------------------------------------------------

# define Z(binary, nonlinear with eta) for group (0,1)

#-----------------------------------------------------------------------


Z.logit        <- rep (0, 2*n )

Zbin                    <- rep (0, 2*n )

#logit scale

Z.logit [1: n]                          <- f1 ( eta [1:n] )

Z.logit [(n+1) : (2*n) ]        <- f2 ( eta [(n+1) : (2*n) ] )


# Simulate.1 binary Z to have Prob(Z = 1) = exp(Z.logit)/(1 + exp(Z.logit))

Z.prob <- exp(Z.logit)/(1 + exp(Z.logit))                       # convert logit to probability

Zbin  <- rbinom (2*n, 1, Z.prob )


BinArray[,,j] <- cbind ( y1, y2, y3, y4, y5, Zbin, G)

BinArray2[,,j] <- cbind ( y1, y2, y3, y4, y5, Zbin, G, eta )

}
```

```
#=============================================

#    ANALYZE BINARY Z: Given observed eta

#=============================================


#-----------------------------------------------

# establish population characteristics graphically

#-----------------------------------------------

plot(density(eta), main="Eta (Population values)")

### LOGIT SCALE

Z.logit.true  <- Z.logit

par(mfrow = c(1, 1))

yrange <- range(Z.logit.true)

xrange <- range(eta)

plot(eta, Z.logit.true, type="n", xlim=xrange,ylim=yrange,

                main = "Observed Z vs True Eta",

                xlab = "eta (true values)", ylab = "Z (logit scale)",

                sub = "Vertical lines at percentiles of eta")

points(eta [1:n], Z.logit.true[1:n], pch=19 , col=4)

points( eta [(n+1) : (2*n) ], Z.logit.true[(n+1) : (2*n) ], col=2)

Q <- matrix(quantile(eta, c(.10, .25, .50, .75, .90)))
```

```
segments(Q[1], yrange[1], Q[1], yrange[2], lty = 2)

segments(Q[2], yrange[1], Q[2], yrange[2], lty = 2)

segments(Q[3], yrange[1], Q[3], yrange[2], lty = 2)

segments(Q[4], yrange[1], Q[4], yrange[2], lty = 2)

segments(Q[5], yrange[1], Q[5], yrange[2], lty = 2)


text( Q[1], yrange[1], " 10th ", , adj = c(0,0), par(srt=90))

text( Q[2], yrange[1], " 25th ", , adj = c(0,0), par(srt=90))

text( Q[3], yrange[1], " 50th ", , adj = c(0,0), par(srt=90))

text( Q[4], yrange[1], " 75th ", , adj = c(0,0), par(srt=90))

text( Q[5], yrange[1], " 90th ", , adj = c(0,0), par(srt=90))


### PROBABILITY SCALE

# Prob(Z = 1) = 1/[1+exp(- Z.logit.true)]

Z.prob.true <- 1/(1+exp(- Z.logit.true))

par(mfrow = c(1, 1))

yrange <- c(-0.1, 1)

xrange <- range(eta)
```

```
# plot fixed values with no error term
 plot(eta, Z.prob.true, type="n", ylim = yrange,
                xlab = "eta (population values)", ylab = "Probability of Z (population)",
                sub = "Vertical lines at percentiles of eta")
points(eta [1:n], Z.prob.true[1:n], pch=19 , col=4)
points( eta [(n+1) : (2*n) ], Z.prob.true[(n+1) : (2*n) ], col=2)
Q <- matrix(quantile(eta, c(.10, .25, .50, .75, .90)))

segments(Q[1], yrange[1], Q[1], yrange[2], lty = 2)
segments(Q[2], yrange[1], Q[2], yrange[2], lty = 2)
segments(Q[3], yrange[1], Q[3], yrange[2], lty = 2)
segments(Q[4], yrange[1], Q[4], yrange[2], lty = 2)
segments(Q[5], yrange[1], Q[5], yrange[2], lty = 2)

text( Q[1], yrange[1], " 10th ", , adj = c(0,0), par(srt=90))
text( Q[2], yrange[1], " 25th ", , adj = c(0,0), par(srt=90))
text( Q[3], yrange[1], " 50th ", , adj = c(0,0), par(srt=90))
text( Q[4], yrange[1], " 75th ", , adj = c(0,0), par(srt=90))
text( Q[5], yrange[1], " 90th ", , adj = c(0,0), par(srt=90))
```

```
#####################################################################
#    SELECT A COPY FROM 50 DATASETS FOR Z-REGRESSION MODEL
#####################################################################


## For Analyses of 1st copy of 50 datasets
dat.copy <- data.frame(BinArray2[, ,1]) #  1st copy
names(dat.copy)
table(dat.copy$GRP)
table(dat.copy$Z)
table(dat.copy$GRP, dat.copy$Z)
# subset dataset for analysis of GAM component of ALV model
dat.copy2 <- data.frame(dat.copy[,6:8])
#-------------------------------------------
#    GAM estimates of Z-population parameters
#-------------------------------------------
etaG <- dat.copy$ETA * dat.copy$GRP
fitZ0.b1      <- gam(Z ~ s(ETA) + GRP + s(etaG), family=binomial, data = dat.copy)
summary(fitZ0.b1)
var(residuals(fitZ0.b1))
pred.y <-  predict(fitZ0.b1,  se = TRUE) # predicted values on logit scale
```

```
#---------------------------------

# ANALYTIC PLOTS

#---------------------------------

lwd<-2; lwd2<-1;

Tx.col<-2; Ctr.col<-4;

fit <- pred.y$fit

UL <- pred.y$fit + pred.y$se.fit

LL <- pred.y$fit - pred.y$se.fit

 group <- G

risk <- eta

ord <- order(risk)

xrange <- range(risk)

yrange <- range(fit)

yrange <- range(Z.logit.true)


plot(risk, fit, type = "n",

#   main=  paste("Variation in Intervention Impact by Baseline Risk"),

      sub = "Vertical lines at percentiles of eta",

   ylim=yrange,

   xlim=xrange,

   xlab = " eta (population values)",
```

```
#   ylab = paste("Probability of Z (GAM fit)"))         # for continous Z
   ylab = paste("fitted Z (GAM)"))              # for binary Z
#   ylab = paste("fitted Z (GLM)"))              # for binary Z


 xord          <- risk[ord]
 fitord        <- fit[ord]
 Grord1        <- group[ord]
 ULord         <- UL[ord]
 LLord         <- LL[ord]


#lines(xord[Grord1 == 1 ],  fitord[Grord1 == 1 ], lty=1, lwd=lwd, col=Tx.col)
#lines(xord[Grord1 == 0 ], fitord[Grord1 == 0 ], lty=2, lwd=lwd, col=Ctr.col)


lines(xord[Grord1 == 1 ],  ULord[Grord1 == 1 ], lty=2, col=Tx.col)
points(xord[Grord1 == 1 ],  fitord[Grord1 == 1 ], col=Tx.col)
lines(xord[Grord1 == 1 ],  LLord[Grord1 == 1 ], lty=2, col=Tx.col)


lines(xord[Grord1 == 0 ],  ULord[Grord1 == 0 ], lty=2, col=Ctr.col)
points(xord[Grord1 == 0 ], fitord[Grord1 == 0 ],  pch=19 , col=Ctr.col)
lines(xord[Grord1 == 0 ],  LLord[Grord1 == 0 ], lty=2, col=Ctr.col)
```

```
#tx.legend <- paste("Tx (n = ", sum(Grord1), ")")

#ctrl.legend <- paste("Ctrl (n = ", sum( (Grord1 == 0) ),")")

#legend(xrange[1],yrange[2], legend = c(tx.legend, ctrl.legend), lty=c(1,2), lwd=c(lwd, lwd),
col=c(Tx.col, Ctr.col))

Q <- matrix(quantile(eta, c(.10, .25, .50, .75, .90)))


segments(Q[1], yrange[1], Q[1], yrange[2], lty = 2)

segments(Q[2], yrange[1], Q[2], yrange[2], lty = 2)

segments(Q[3], yrange[1], Q[3], yrange[2], lty = 2)

segments(Q[4], yrange[1], Q[4], yrange[2], lty = 2)

segments(Q[5], yrange[1], Q[5], yrange[2], lty = 2)


text( Q[1], yrange[1], " 10th ", , adj = c(0,0), par(srt=90))

text( Q[2], yrange[1], " 25th ", , adj = c(0,0), par(srt=90))

text( Q[3], yrange[1], " 50th ", , adj = c(0,0), par(srt=90))

text( Q[4], yrange[1], " 75th ", , adj = c(0,0), par(srt=90))

text( Q[5], yrange[1], " 90th ", , adj = c(0,0), par(srt=90))
```

```
#=============================================

# save simulated datasets for later replication studies

#=============================================


write.csv(ContArray, file = "C:/.../repDat.N300.ContLin.csv", row.names = FALSE)

write.csv(BinArray, file = "C:/.../repDat.N300.BinNlin.csv", row.names = FALSE)

write.csv(ContArray2, file = "C:/.../repDat2.N300.ContLin.csv", row.names = FALSE)

write.csv(BinArray2, file = "C:/.../repDat2.N300.BinNlin.csv", row.names = FALSE)
```

APPENDIX B: SELF-WRITTEN R FUNCTIONS CALLED BY ALV MODEL

```
############################################################
#    R-FUNCTIONS FOR THE GAM COMPONENT OF ALV MODEL
############################################################
# LIST OF FUNCTIONS
#  (1) write R function to define R(x,z)for cubic spline on [0,1]
#              function name = rk
#  (2) Use the rk function in a new function that takes a sequence of knots
#    and an array of x values to produce a model matrix X for cubic spline (p127)
#              function name = spl.X
#  (3) write a function to setup a penalized regression spline penalty matrix S
#              function name = spl.S
#  (4) write a simple matrix sqrt function to use on S
#              function name = mat.sqrt
#  (5) write a function to SET UP a simple additive model
#    with 2 smooth terms + 1 parametric term. This function is modified from the
#    function am.setup(Wood, 2006, p 135) and calls functions (1) to (3).
#              function name = am.setup2
```

APPENDIX B: (CONTINUED)

```
#####################################
#    FOR GAM COMPONENT
#####################################


# SIMPLE CUBIC SPLINE
# write R function to define R(x,z)for cubic spline on [0,1]
rk <- function(x,z)
    {       ((z - 0.5)^2 - 1/12)*((x - 0.5)^2 - 1/12)/4 -
            ((abs(x - z) - 0.5)^4 - (abs(x - z) - 0.5)^2/2 + 7/240)/24
    }
# Use the rk function to write a function that takes a sequence of knots
#  and an array of x values to produce a model matrix X for cubic spline (p127)
spl.X <- function(x,xk)
    {       q       <- length(xk) + 2               # number of params
            n       <- length(x)            # number of data
            X       <- matrix(1, n, q)      # initialize model matrix
            X[,2] <- x                              # set 2nd column to x
            X[,3:q] <- outer(x,xk,FUN=rk) # and remaining to R(x,xk)
            X
    }
```

```
# EXTENSION TO PENALIZED CUBIC SPLINE


# Model extension: to fit penalized regression spline to  x, y, data
# First write a function to setup a penalized regression spline penalty matrix S
spl.S <- function(xk)            # i.e. given a knot sequence xk
    {
            q <- length(xk) + 2; S <- matrix(0,q,q)    # init S to 0
            S[3:q, 3:q] <- outer(xk,xk,FUN=rk)                  # fill in nonzero part
            S
    }
# need a simple matrix sqrt function to use on S
mat.sqrt <- function(S)
    {
            d <- eigen(S, symmetric = TRUE)
            rS <- d$vectors%*%diag(d$values^0.5)%*%t(d$vectors)
    }
```

```
# EXTENSION TO ADDITIVE MODEL

# Write a function to SET UP a 3-term simple additive model :

# function to produce a model matrix X

#     and 2 regression penalty matrices in S for

#     a 2-smooth + 1-parametric terms additive model

am.setup2 <- function(x, z, g, q = 10)

    # get X, S_1 and S_2  for a simple 2-term (x & z) AM

    #          including 1 parametric term g

    {

            # choose knots

            xk <- quantile(unique(x), 1:(q-2)/(q-1))

            zk <- quantile(unique(z), 1:(q-2)/(q-1))

            # get penalty matrices

            S <- list()

            S[[1]]   <- S[[2]]          <- matrix(0, 2*q, 2*q)

            S[[1]][3:(q+1), 3:(q+1)]                    <- spl.S(xk)[-1, -1]

            S[[2]][(q+2):(2*q), (q+2):(2*q)]            <- spl.S(zk)[-1, -1]


            # get model matrix the 2 smooth terms

            n <- length(x)

            X1 <- matrix(1, n, 2*q-1)
```

```
        X1[ ,2:q]                       <- spl.X(x, xk)[ , -1]      # 1st smooth
        X1[ ,(q+1):(2*q-1)]       <- spl.X(z, zk)[ , -1]      # 2nd smooth


        # add parametric term to 2nd column of model matrix
        d <- dim(X1)[2]
        X <- cbind(X1[,1], g, X1[,2:d])
        dimnames(X) <- NULL
        list(X=X, S=S)
}
```

```
#    FOR MCMC ALGORITHM

# Define the unnormalized log-density of the cond distribution of eta

#    from which to draw a sample.

# The function accepts data from the ith independent observation.


condETAfun.gam <- function(eta_i, YZ, muY, lambda, theta, beta, sigma.sq, Xmat_i,
penalty)

        {

                Y <- matrix(YZ[c(1:p)], ncol=1)

                Z <- matrix(YZ[p+1])

                # Allow ith eta to alternate btw candidate and current values

                #          so that both values contribute to its condit distrib in turns:

                # Note - eta is in 3rd column of model matrix

                Xmat_i[3]  <- eta_i        # insert eta value (when eta = canditate/current)

                # Define cond distrib of eta_i upto a constant

                logLik <-          (

               -0.5 %*% t(Y-muY-lambda%*%eta_i) %*% solve(theta) %*% (Y-muY-
lambda%*%eta_i)        +

                -0.5 * 1/sigma.sq * ((Z - t(Xmat_i) %*% beta)^2 )          +

                -0.5 * eta_i^2

                        )

        }
```

APPENDIX C: R CODES FOR ALV MODEL ALGORITHM

```
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
#	MUST FIRST RUN ALV FUNCTIONS IN R (APPENDIX B)
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
# load libraries.
library(foreign)
library(mgcv)
library(nlme)
library(MASS)
library(MCMCpack)
library(numDeriv


#	GET SIMULATED DATASETS (50 COPIES STACKED HORIZONTALLY)
replicData <- read.csv("C:/.../repDat.N300.ContLin.csv", header = TRUE)
dim(replicData)
#---------VARIABLE LABELS FOR DATASET----------
#	Y1-Y5		= continous scale indicators
#	Z		= binary/cont distal outcome
#	GRP		= 2-level group
#-----------------------------------------------
```

```
### assign GLM family of distribution

Y1.family        <- gaussian

Y2.family        <- gaussian

Y3.family        <- gaussian

Y4.family        <- gaussian

Y5.family        <- gaussian

#---- select distribution for  Z ----

z.binary <- TRUE

Z.family <- binomial

# OR

z.binary <- FALSE

Z.family <- gaussian

#### USE THE 1ST COPY OF REPLIC DATASETS TO INITIATE SOME PARAMETER
VALUES

YZdata <- data.frame(replicData[,1:7])

names(YZdata)  <- c("Y1","Y2","Y3","Y4","Y5","Z","GRP")

#dim(YZdata)

N <- nrow(YZdata)

p <- ncol(YZdata)-2       # let p = dim YZdata less (Z, GRP) -> no of Y variables

Ydata <- YZdata[ ,1:p]

Zdata <- YZdata[ ,(p+1)]
```

```
#----------------------------------------
#          TRUE VALUES
#----------------------------------------
# STORE TRUE VALUES (WHERE AVAILABLE) FOR EASY TABULATION
# Y's
muY.t              <- matrix(rep(0, p), ncol = 1)
muY.se.t           <- matrix(rep(NA, p), ncol = 1)
muY.pval.t         <- matrix(rep(NA, p), ncol = 1)
Rsq.muY.t          <- matrix(rep(NA, p), ncol = 1)

lambda.t           <- matrix(rep(1, p), ncol = 1)
lambda.se.t        <- matrix(rep(NA, p), ncol = 1)
lambda.pval.t      <- matrix(rep(NA, p), ncol = 1)

theta.t            <- diag(rep(0.25,p))
theta.se.t         <- matrix(rep(NA, p), ncol = 1)

# Z
muZ.t              <- matrix(0)
muZ.se.t           <- matrix(NA)
muZ.pval.t         <- NA
Rsq.Z.t            <- NA
```

```
beta.t              <- matrix(NA)

beta.se.t           <- matrix(NA)

beta.pval.t         <- NA


sigma.sq.t          <- matrix(0.25)

sigma.sq.se.t       <- matrix(NA)


grp.interc.t        <- matrix(NA)

grp.interc.se.t     <- matrix(NA)

grp.interc.pval.t   <- NA


etaBYgrp.t          <- matrix(NA)

etaBYgrp0.se.t      <- matrix(NA)

etaBYgrp0.pval.t    <- NA
```

```
#----------------------------------------
#          START VALUES
#----------------------------------------
muY0          <- matrix(apply(YZdata[ ,1:p], 2, mean), ncol = 1)
muY0.se       <- matrix(rep(NA, p), ncol = 1)
muY0.pval     <- matrix(rep(NA, p), ncol = 1)
Rsq.muY0      <- matrix(rep(NA, p), ncol = 1)

lambda0       <- matrix(rep(0.5,p), ncol = 1)
lambda0.se    <- matrix(rep(NA, p), ncol = 1)
lambda0.pval  <- matrix(rep(NA, p), ncol = 1)

#theta0       <- diag(rep(1,p))
theta0        <- diag(apply(YZdata[ ,1:p], 2, var)) # standard for ALV
theta0.se     <- matrix(rep(NA, p), ncol = 1)

# obtain approx z-interc when regressed on GRP + Y1 + Y2 + Y3 + Y4 + Y5
muZ0          <- matrix(glm(Z ~ . ,family = Z.family, data=YZdata)$coefficients[1])
muZ0.se       <- NA
muZ0.pval     <- NA
Rsq.Z0        <- NA
```

```
beta0    <- NA # initial spline coefficients to be estimated shortly
#beta0.se          <- matrix(NA)

#sigma.sq0         <- matrix(1.0)
sigma.sq0          <- matrix(var(YZdata[ ,(p+1)])) # standard for ALV

etaBYgrp0                  <- matrix(0)
etaBYgrp0.se     <- NA
etaBYgrp0.pval <- NA

grp.interc0        <-  matrix(1)
grp.interc0.se     <-  NA
grp.interc0.pval <-  NA

#--------------------------------
### start values for eta
#--------------------------------
# Compute approx var(eta|Y,Z,current params) from the start values
B <- solve((lambda0 %*% t(lambda0)+ theta0))
sigma2.eta0 <- 1 - t(lambda0) %*% B %*% lambda0
#sigma2.eta0
```

```
# generate initial N-vector eta from its cond distrib
eta00 <- matrix(rep(NA, N))
for (i in 1:N)
        {
                Yi <- matrix(as.numeric(Ydata[i, ], ncol=1))
                eta00[i] <- t(lambda0) %*% B %*%(muY0-Yi)
        }
#plot(density(eta00))
# GENERATE INITIAL PENALIZED REGRESSION SPLINE MODEL MATRIX
# Scale eta00 to lie in [0,1]
        x       <- eta00 - min(eta00); x <- x/max(x)
# Next select a rank=30 basis (a set of 28 knots evenly spread over [0,1];
        xk      <- 1:28/29              # choose knots
        q       <- length(xk) + 2       # dimension of basis
# Call function to produce model matrix
Xmat0 <- spl.X(x, xk)
Smat0 <- spl.S(xk)
# NEXT GENERATE INITIAL ESTIM OF SPLINE COEFF AND PENALTY TERM
fit     <- gam(Z ~ s(eta00, GRP, k=q) , family=Z.family, data=YZdata)
#summary(fit)
beta0 <- matrix(fit$coefficients, ncol=1)
tau0    <- fit$gcv.ubre
penalty0        <- tau0 * (t(beta0) %*% Smat0 %*% beta0 )
```

APPENDIX C: (CONTINUED)

```
#####################################################
#                  ALV (MCEM) ALGORITHM
#####################################################


#------------------------------------------------------------
# SET PARAMETERS FOR ALV ALGORITHM
#------------------------------------------------------------
# NOTE:
#         SINGLE REPLICATION TO STUDY CONVERGENCE (set bridge = TRUE)
#         MULTIPLE REPLICATIONS TO STUDY ESTIMATION (set bridge = FALSE)


# Start RUN from here
        bridge   <- TRUE
        maxrep <- 50    # of datasets to analyze
        stop.iter<-  3     # number of EM iterations following min deviance.
                                #(set to 100 for convergence studies)
maxiter         <- 100  # set maximum EM iterations
tuneSize        <- 2.5   # rejection/acceptance control
burninSize      <- 100  # Markov samples in burn-in period (to discard)
mcmcSize        <- 100  # Length of MCMC chain retained for analysis
thinSize        <- 1      # for thinning size
M <- mcmcSize/thinSize  # Effective length of Markov chain used in analysis remains constant
```

APPENDIX C: (CONTINUED)

```
#############################################################################
#==========          START OF ALV MODEL RUN                    ============
#############################################################################
## intialize lines as pointers for tracking different stages in the MCEM loop
line1 <- 0; line2 <- 0; line3 <- 0; line4 <- 0; line5 <- 0;
line6 <- 0; line7 <- 0; line8 <- 0; line9 <- 0; line10 <- 0
#---------------------------------------------------------------------------
####  Initialize storage matrices for all MCEM replications
#---------------------------------------------------------------------------
etaVectors <- matrix(NA, N, maxrep)
Z.best.mat <- matrix(NA, N, maxrep)

# means and variances of ESTIMATES
paramMeans1 <- matrix(NA, (3*p+3), maxrep)  # for non-smoothed param est
paramMeans2 <- matrix(NA, (q+1), maxrep)              # for q spline coeff + 1 column
paramVars1      <- matrix(NA, (3*p+3), maxrep)
paramVars2      <- matrix(NA, (q+1), maxrep)
# means and variances of ESTIMATES
paramMeans1.se          <- matrix(NA, (3*p+3), maxrep)          # for non-smoothed param est
paramMeans2.se          <- matrix(NA, (q+1), maxrep)            # for spline coeff
paramVars1.se  <- matrix(NA, (3*p+3), maxrep)
paramVars2.se  <- matrix(NA, (q+1), maxrep)
```

```
# sub-model deviances
all.deviances     <- matrix(NA, maxrep, 6)


#---------------------------------------------------------
#  START ALV REPLICATIONS: 1ST LOOP
#---------------------------------------------------------
# clock the start of EM iterations
Start.time <- Sys.time()


replic    <- 0
while ( maxrep > replic )
{
          replic <- replic + 1
# get a copy from 50 replicate datasets stacked horizontally
#          (7 variable columns per dataset)
r                    <- replic                    # for rth dataset; r=1,...,50
d                    <- replicData[ , ((r-1)* 7 + 1):(r * 7)] # select the rth 7 columns for rth dataset
YZdata <- data.frame(d)
Ydata <- YZdata[ ,1:p]
Zdata <- YZdata[ ,(p+1)]
names(YZdata)  <- c("Y1","Y2","Y3","Y4","Y5","Z","GRP")
```

APPENDIX C: (CONTINUED)

```
#### create storage matrices for kth MCEM iteration

#  arrays to store calculated MEANS of individual regression parameter values


Y_params                <- matrix(0, nrow=(maxiter+2), ncol=(10*p))

dim(Y_params)           <- c((maxiter+2), 10, p)


dimnames(Y_params)<- list(NULL, c("EM-iter","interc","(s.e.)","p.value","lambda","(s.e.)",
                                                    "p.value","theta","R^2","deviance"),
names(Ydata))

Y_params[ ,1,]  <- c("true", "start", c(1:maxiter)) # input EM counter index


Z_params1                       <- matrix(0, nrow=(maxiter+2), ncol=13)

dimnames(Z_params1)  <- list(NULL, c("EM-
iter","threshold","(s.e.)","p.value","beta","(s.e.)","p.value",
"grp","(s.e.)","p.value","sigma^2","dev_explained","deviance"))

Z_params1[ ,1]          <- c("true", "start", c(1:maxiter)) # input EM counter index

Z_params2                       <- matrix(0, nrow=(maxiter+1), ncol=(q+1))

dimnames(Z_params2)  <- list(NULL, c("EM-iter","interc",rep("s(eta.grp)", (q-1))))

Z_params2[ ,1]          <- c("start", c(1:maxiter)) # input EM counter index

Z_params3                       <- matrix(NA, nrow=(maxiter), ncol=2)

dimnames(Z_params3)  <- list(NULL, c("EM-iter","UBRE score (tau)"))

Z_params3[ ,1]          <- c(1:maxiter)                                 # input EM
counter index
```

APPENDIX C: (CONTINUED)

```
#  arrays to store calculated VARIANCES of individual regression parameter values

Y_parVars                <- matrix(0, nrow=(maxiter+2), ncol=(10*p))

dim(Y_parVars)           <- c((maxiter+2), 10, p)

dimnames(Y_parVars)<- list(NULL, c("EM-iter","interc","(s.e.)","p.value","lambda","(s.e.)",

"p.value","theta","R^2","deviance"), names(Ydata))

Y_parVars[ ,1,] <- c("true", "start", c(1:maxiter)) # input EM counter index


Z_parVars1                       <- matrix(0, nrow=(maxiter+2), ncol=13)

dimnames(Z_parVars1) <- list(NULL, c("EM-
iter","threshold","(s.e.)","p.value","beta","(s.e.)","p.value",

"grp","(s.e.)","p.value","sigma^2","dev_explained","deviance"))

Z_parVars1[ ,1]          <- c("true", "start", c(1:maxiter)) # input EM counter index


# RECORD the true values in first row of parameters table

for (h in 1:p)

        {

  Y_params[1 ,c(2:10), h] <- round(cbind(muY.t[h],
muY.se.t[h],muY.pval.t[h],lambda.t[h],lambda.se.t[h],

                                        lambda.pval.t[h], theta.t[h,h], Rsq.muY.t[h], NA), 4)

        }

  Z_params1[1, c(2:13)] <- round(cbind(muZ.t, muZ.se.t, muZ.pval.t, beta.t, beta.se.t, beta.pval.t,

                                        grp.interc.t, grp.interc.se.t, grp.interc.pval.t,

                                         sigma.sq.t, Rsq.Z.t, NA), 4)
```

```
# Store initial values in 2nd row of parameters table

for (h in 1:p)

        {

  Y_params[2 ,c(2:10), h] <- round(cbind(muY0[h], muY0.se[h], muY0.pval[h], lambda0[h],
lambda0.se[h],

                                        lambda0.pval[h], theta0[h,h], Rsq.muY0[h], NA), 4)

        }

  Z_params1[2, c(2:13)] <- round(cbind(muZ0, muZ0.se, muZ0.pval, NA, NA, NA,

                                        grp.interc0, grp.interc0.se, grp.interc0.pval,

                                        sigma.sq0, Rsq.Z0, NA), 4)

  Z_params1[2, 5]               <- "spline"

  Z_params2[1, c(2:(q+1))] <- round(as.vector(beta0), 4)


# initialize storage of best MCEM output results

iter.best               <- 0

minDeviance             <- 0


#### create a matrix to store deviance & convergence values for MCEM

convergence             <- data.frame(matrix(NA,nrow=maxiter, ncol=7))

convergence[, 1]                <- 1:maxiter

convergence[, 2]                <- 999999

names(convergence)      <- c("EM-iter", "SumDeviance", "Conv.Err1",

                                        "Conv.Err2", "Conv.Err", "logLR.com", "logLR.obs")
```

```
log_LR <- matrix(NA, nrow=1, ncol=maxiter)

Y.estim.se <- matrix(0, maxiter*6*p)
dim(Y.estim.se)          <- c(maxiter, 6, p)

Z.estim.se <- matrix(0, maxiter, 6)

# Matrix to store std error estim by Louis method
louis.se <- matrix(NA, nrow=20, ncol=maxiter)

#-----------------------------------------------------------------------
#       START MCEM ITERATIONS (2ND LOOP)
#-----------------------------------------------------------------------
## Get start values
muY             <- muY0
lambda <- lambda0
theta    <- theta0
muZ             <- muZ0
beta            <- beta0
sigma.sq        <- sigma.sq0
Xmat            <- Xmat0
penalty <- penalty0
```

APPENDIX C: (CONTINUED)

```
# supply initial parameter values
new.params1     <- c(muY, lambda, theta, sigma.sq)
new.params2     <- c(beta)
new.params       <- c(muY, lambda, theta, beta, sigma.sq)
iter                  <- 0
while(  maxiter > iter )
{
  iter     <- iter + 1                                          # update EM counter
# to store eta statistics for N subjects
eta.chains          <- matrix(0, nrow=N, ncol= M) # to store N Markov chains
eta.stat             <- matrix(0, nrow=N, ncol=4) # initialize matrix to store eta statistics
eta.stat             <- data.frame(eta.stat)
names(eta.stat)  <- c("Mean", "SD", "Naive SE", "Time-series SE")
# create matrices to record bridge sampling results
Lik_aa <- matrix(NA, nrow=N, ncol=M)
Lik_ab <- matrix(NA, nrow=N, ncol=M)
Lik_ba <- matrix(NA, nrow=N, ncol=M)
Lik_bb <- matrix(NA, nrow=N, ncol=M)
# create matrices to record parameter values to be generated in the current EM iteration
Yparam.est                  <- matrix(0, nrow=M, ncol=(9*p))
dim(Yparam.est)          <- c(M, 9, p)
Zparam.est1      <- matrix(0, nrow = M, ncol=12)
Zparam.est2      <- matrix(0, nrow = M, ncol=(q+1))
```

```
# create an array to record derivatives and calculated stderr to be generated in
#          the current EM iteration using Louis' formula


############
# E-step
############
#-----------------------------------------------------------
#  START M-H ITERATION: 3RD LOOP
#-----------------------------------------------------------
old.params1 <- new.params1      # save current parameter values
old.params2 <- new.params2
old.params <- new.params
## Metropolis-Hastings algorithm is performed on each subject
#          to simulate from p(eta|observed data at current values)
 line1   <- line1 + 1
 # sample within GRP level: YZdata is sorted by GRP
  a <- round(N/2)
 samp1 <- sample(1:a, 5, replace = FALSE)
 samp2 <- sample((a+1):N, 5, replace = FALSE)
 samp  <- c(samp1, samp2)
```

```
 # SUBJECT loop to run N Markov chains (1 for each subject i = 1:N)
i       <- 0            # initialize subject (row) counter
 while ( N > i )
{       # begin M-H inner loop
        i <- i + 1
        YZi             <- as.numeric( YZdata[i, ] )            # select ith observed data row
        eta_i   <- Xmat[i, 3]                                   # eta is in 3rd column of model matrix
        Xmat_i <- matrix( Xmat[i, ], ncol=1 )
        count <- 0
        repeat
        {
        count <- count + 1
        testrun <- try(MCMCmetrop1R(condETAfun.gam, theta.init= eta_i, Xmat_i=Xmat_i,
YZ=YZi,  muY=muY, theta=theta,  lambda=lambda, penalty=penalty, beta=beta,
sigma.sq=sigma.sq, thin=thinSize, mcmc=mcmcSize, burnin=burninSize, tune=tuneSize,
seed=NA,  optim.method = "BFGS", verbose=0, logfun=TRUE, force.samp = TRUE,
optim.control = list(fnscale = -1, trace = 0, REPORT = 10, maxit=1000) ))
                if (class(testrun) != "try-error" || count > 5) break
        }
  eta.samp <- testrun
####    SCRIPT FOR EXAMINING MCMC OPTIMALITY    ####
# plot(eta.samp)
#  raftery.diag(eta.samp)
```

```
#raftery <- raftery.diag(eta.samp)
#ifelse (raftery$resmatrix[1] < 30, burninSize <- 50, burninSize <- 100)
####     END OF SCRIPT FOR EXAMINING MCMC OPTIMALITY    ####


eta.chains[i, ] <- t(eta.samp)
 eta.stat[i, ]        <- summary(eta.samp)$statistics
line2 <- line2 + 1
# store MCMC samples of 5 randomly selected observations (subject) for diagnostics
 if (i == samp[1]) eta.samp1.1 <- eta.samp
 if (i == samp[2]) eta.samp1.2 <- eta.samp
 if (i == samp[3]) eta.samp1.3 <- eta.samp
 if (i == samp[4]) eta.samp1.4 <- eta.samp
 if (i == samp[5]) eta.samp1.5 <- eta.samp
 if (i == samp[6]) eta.samp2.1 <- eta.samp
 if (i == samp[7]) eta.samp2.2 <- eta.samp
 if (i == samp[8]) eta.samp2.3 <- eta.samp
 if (i == samp[9]) eta.samp2.4 <- eta.samp
 if (i == samp[10]) eta.samp2.5 <- eta.samp
#-----------------------------------------------------
#  END M-H ITERATION: 3RD LOOP
#-----------------------------------------------------
}
```

```
##########
# M-step:  Estimate new parameters given the expected value of eta
##########
#          linear regression of Y's on etaHat
### Personal note: This loop will be generalized later to accept any number p of regressions
line3 <- line3 + 1
for (j in 1:M )
        {
        # FIT Y INDICATORS
        eta_j           <-      as.numeric(eta.chains[ ,j])
        fitY1   <- glm(YZdata[ ,1] ~ eta_j, family = Y1.family)
        fitY2   <- glm(YZdata[ ,2] ~ eta_j, family = Y2.family)
        fitY3   <- glm(YZdata[ ,3] ~ eta_j, family = Y3.family)
        fitY4   <- glm(YZdata[ ,4] ~ eta_j, family = Y4.family)
        fitY5   <- glm(YZdata[ ,5] ~ eta_j, family = Y5.family)

        Yparam.est[j, 1:3, 1]   <- summary(fitY1)$coefficients[1, c(1,2,4)]     # extract interc,
s.e., p-value
        Yparam.est[j, 4:6, 1]   <- summary(fitY1)$coefficients[2, c(1,2,4)]     # extract slope,
s.e., p-value
        Yparam.est[j, 7, 1]     <- var(residuals(fitY1))
        Yparam.est[j, 8, 1]     <- NA # place holder for R-squared
        Yparam.est[j, 9, 1]     <- fitY1$deviance
```

```
        Yparam.est[j, 1:3, 2]   <- summary(fitY2)$coefficients[1, c(1,2,4)]   # extract interc,
s.e., p-value

        Yparam.est[j, 4:6, 2]   <- summary(fitY2)$coefficients[2, c(1,2,4)]   # extract slope,
s.e., p-value

        Yparam.est[j, 7, 2]     <- var(residuals(fitY2))

        Yparam.est[j, 8, 2]     <- NA # place holder for R-squared

        Yparam.est[j, 9, 2]     <- fitY2$deviance


        Yparam.est[j, 1:3, 3]   <- summary(fitY3)$coefficients[1, c(1,2,4)]   # extract interc,
s.e., p-value

        Yparam.est[j, 4:6, 3]   <- summary(fitY3)$coefficients[2, c(1,2,4)]   # extract slope,
s.e., p-value

        Yparam.est[j, 7, 3]     <- var(residuals(fitY3))

        Yparam.est[j, 8, 3]     <- NA # place holder for R-squared

        Yparam.est[j, 9, 3]     <- fitY3$deviance


        Yparam.est[j, 1:3, 4]   <- summary(fitY4)$coefficients[1, c(1,2,4)]   # extract interc,
s.e., p-value

        Yparam.est[j, 4:6, 4]   <- summary(fitY4)$coefficients[2, c(1,2,4)]   # extract slope,
s.e., p-value

        Yparam.est[j, 7, 4]     <- var(residuals(fitY4))

        Yparam.est[j, 8, 4]     <- NA # place holder for R-squared

        Yparam.est[j, 9, 4]     <- fitY4$deviance


        Yparam.est[j, 1:3, 5]   <- summary(fitY5)$coefficients[1, c(1,2,4)]   # extract interc,
s.e., p-value
```

```
        Yparam.est[j, 4:6, 5]     <- summary(fitY5)$coefficients[2, c(1,2,4)]       # extract slope,
s.e., p-value

        Yparam.est[j, 7, 5]       <- var(residuals(fitY5))

        Yparam.est[j, 8, 5]       <- NA # place holder for R-squared

        Yparam.est[j, 9, 5]       <- fitY5$deviance


line4 <- line4 + 1


        # FIT DISTAL OUTCOME Z

        fitZ      <- gam(Z ~ s(eta_j, GRP, k=q) , family = Z.family, data=YZdata)


        Zparam.est1[j, 1:3]       <- summary(fitZ)$p.table[1 ,c(1,2,4)]     # extract interc, s.e., p-
value

        Zparam.est1[j, 4:6]       <- c(NA, NA, NA)                                    # beta's
not recorded here

        Zparam.est1[j, 7:9]       <- c(NA, NA, NA)

        Zparam.est1[j, 10]        <- var(residuals(fitZ))

        Zparam.est1[j, 11]        <- summary(fitZ)$dev.expl       # extract deviance explained

        Zparam.est1[j, 12]        <- fitZ$deviance

        Zparam.est2[j, 1:q]       <- fitZ$coefficients       # extract spline coeff

        Zparam.est2[j, (q+1)]     <- fitZ$gcv.ubre   # extract estimated smoothing parameter tau

        }


line5 <- line5 + 1
```

```
#############################################################################
### Calculate the means & Monte Carlo std err of parameter estimates for current EM iteration
#############################################################################

#===============
#  parameters for Y
#================
Ymeans          <- matrix(NA, nrow = p, ncol = 9)
Yvars           <- matrix(NA, nrow = p, ncol = 9)

for (k in 1:p)
 {
        ## calculate col means/variances of kth array in Yparam.est, form a vector
        #       store temporarily
        Ymeans[k, ] <- matrix(apply(Yparam.est[ , , k], 2, mean), nrow=1)
        Yvars[k, ] <- matrix(apply(Yparam.est[ , , k], 2, var), nrow=1)

        # store results for MEANS permanently in kth array in Y.estim.se
        Y_params[(iter+2), 2:10, k]       <- round(Ymeans[k, ], 4)
```

```
        # Record y-interc, lambdas, thetas and calculate their MC std err of estimates

        Y.estim.se[iter, c(1,3,5), k]       <- round(Ymeans[k, c(1,4,7)], 3)

        Y.estim.se[iter, 2, k]      <- round(sqrt((sum((Yparam.est[ , 1, k] - Ymeans[k,
1])^2))*(1/(M*(M-1)))), 3)

        Y.estim.se[iter, 4, k]      <- round(sqrt((sum((Yparam.est[ , 4, k] - Ymeans[k,
4])^2))*(1/(M*(M-1)))), 3)

        Y.estim.se[iter, 6, k]      <- round(sqrt((sum((Yparam.est[ , 7, k] - Ymeans[k,
7])^2))*(1/(M*(M-1)))), 3)

 }


#======================

# 1st set of parameters for Z

#=======================

# Record Z-threshold, grp-coef, sigma.sq and calculate their MC std err of estimates

Zmeans1 <- matrix(apply(Zparam.est1, 2, mean), nrow=1) #  calculate col means

Zvars1 <- matrix(apply(Zparam.est1, 2, var), nrow=1) #  calculate col variances

Z.estim.se[iter, c(1,3,5)]<- round(Zmeans1[c(1,7,10)], 3)

Z.estim.se[iter, 2]                 <- round(sqrt((sum((Zparam.est1[ , 1] -
Zmeans1[1])^2))*(1/(M*(M-1)))), 3)

Z.estim.se[iter, 4]                 <- round(sqrt((sum((Zparam.est1[ , 7] -
Zmeans1[1])^2))*(1/(M*(M-1)))), 3)

Z.estim.se[iter, 6]                 <- round(sqrt((sum((Zparam.est1[ , 10] -
Zmeans1[1])^2))*(1/(M*(M-1)))), 3)

Z_params1[(iter+2), 2:13]       <- round(Zmeans1, 3)

Z_params1[(iter+2), 5]          <- "spline"
```

```
#==================================

# 2nd set of parameters for Z

#==================================


Zmeans2          <- matrix(apply(Zparam.est2, 2, mean), nrow=1) #  calculate col means

Zvars2  <- matrix(apply(Zparam.est2, 2, var), nrow=1) #  calculate col variances

Z_params2[(iter+1), 2:(q+1)]     <- round(Zmeans2[1:q], 3)  # extract 20 coeff (less ubre score)

Z_params3[(iter), 2]             <- round(Zmeans2[(q+1)], 3) # store gcv.ubre score


line6 <- line6 + 1


##############################################

#          UPDATE parameters for next EM round

##############################################

#------------------------

# Update Y parameters

#------------------------

muY              <- matrix(c(Ymeans[1:p, 1]), ncol=1)

lambda <- matrix(c(Ymeans[1:p, 4]), ncol=1)

theta     <- diag(c(Ymeans[1:p, 7]))
```

```
#------------------------
# Update Z parameters
#------------------------
muZ             <- matrix(Zmeans1[1])              #  threshold/intercept
sigma.sq        <- matrix(Zmeans1[10])
beta            <- matrix(Zmeans2[1:q], ncol=1)    # spline coefficients

tau             <- matrix(Zmeans2[(q+1)])          # smoothing parameter

line7 <- line7 + 1

# TO UPDATE  matrices X, S and penalty: first obtain an N-vector eta from MCMC simulations
eta.vec  <- as.vector(apply(eta.chains, 1, mean))  # get row means (eta Hat for each subject)

# Scale eta.vec to lie in [0,1]
x2      <- eta.vec - min(eta.vec); x2 <- x2/max(x2)
# Call function to produce new model and penalty matrices
Xmat <- spl.X(x2, xk)
Smat <- spl.S(xk)
#       dim(Smat)                  # q x q  penalty matrix for s(eta,grp)
#       dim(Xmat)                  # Nxq model matrix
#       dim(beta)                  # qx1 penalized least sq estimates of spline coefficients
#       dim(tau)                   # scalar : estimate of common smoothing parameter
```

```
# update current estim of penalty of the penalized least square expression for
#          (Z|eta, grp, eta*grp) component of (eta|Y,Z,omega(k)). NOTE: This step is not neccssary
penalty <- tau * (t(beta) %*% Smat %*% beta )


line8 <- line8 + 1


#####################################
#  COMPUTE LOUIS' STD ERRORS
#####################################
#-------------------------------------------------------------
# calculate        partial derivatives w.r.t. muY, lambda and theta
#-------------------------------------------------------------
louis1   <- matrix(0, nrow=M, ncol=(6*p))
dim(louis1) <- c(M, 6, p)
for (j in 1:M )
{
eta_j    <- as.numeric(eta.chains[ ,j])
          for (k in 1:p)
          {
mu       <- Yparam.est[j,1,k]
lam      <- Yparam.est[j,4,k]
the      <- Yparam.est[j,7,k]
y        <- YZdata[ ,k]
```

```
## Calculate the gradient/Hessian of a function by numerical approximation using numDeriv-package functions

func.mu <- function(mu){ -0.5*N*log(the)-(0.5/the)*(sum( (y - mu - lam*eta_j)^2)) }

func.lam <- function(lam){ -0.5*N*log(the)-(0.5/the)*(sum( (y - mu - lam*eta_j)^2)) }

func.the <- function(the){ -0.5*N*log(the)-(0.5/the)*(sum( (y - mu - lam*eta_j)^2)) }


# Store 1st partial derivatives

louis1[j,1,k] <- grad(func.mu, mu          # muY

louis1[j,2,k] <- grad(func.lam, lam)       # lambda

louis1[j,3,k] <- grad(func.the, the)       # theta


# 2nd partial derivatives

louis1[j,4,k] <-  as.double(hessian(func.mu, mu))          # muY

louis1[j,5,k] <-  as.double(hessian(func.lam, lam))        # lambda

louis1[j,6,k] <-  as.double(hessian(func.the, the))# theta

          }

}
```

```
# calculate Louis std err (use NEGATIVE 2nd partial derivatives)

#          Store stacked in a column per EM iter


for (k in 1:p)

{

louis.se[k, iter]  <-  round((mean(( louis1[,1,k] - mean(louis1[,1,k]) )^2) -  mean(louis1[,4,k])), 3)
# muY

louis.se[(k+p), iter]        <-  round((mean(( louis1[,2,k] - mean(louis1[,2,k]) )^2) -
mean(louis1[,5,k])), 3) # lambda

louis.se[(k+2*p), iter] <-  round((mean(( louis1[,3,k] - mean(louis1[,3,k]) )^2) -
mean(louis1[,6,k])), 3) # theta

}



################################################

# MONITOR CONVERGENCE 1 : STANDARD APPROACH

################################################

# Store new parameters

new.params1     <- c(muY, lambda, theta, sigma.sq)

new.params2     <- c(beta)

new.params                  <- c(muY, lambda, theta, beta, sigma.sq)
```

```
# Calculate and update convergence error
err1            <- sqrt(sum((old.params1 - new.params1)^2))
err2            <- sqrt(sum((old.params2 - new.params2)^2))
err             <- sqrt(sum((old.params - new.params)^2))

# Calculate and update total deviance
y.dev <- matrix(1:p, nrow=1)
for (h in 1:p)
        {       y.dev[h]        <- as.numeric(Y_params[(iter+2), 10, h])        }
dev.Z   <- as.numeric(Z_params1[(iter+2), 13])
convergence[iter, 1]    <- iter
convergence[iter, 2]    <- sum(y.dev, dev.Z )
convergence[iter, 3]    <- err1
convergence[iter, 4]    <- err2
convergence[iter, 5]    <- err

# Record and update model fits & MCMC samples for best EM iteration
#       based on minimum total deviance
new.minDeviance         <- min(convergence$SumDeviance)
best <- convergence[convergence$SumDeviance == new.minDeviance, ]
                iter.best       <- as.numeric(best[1])
```

```
if ( iter == iter.best )
        {
        eta.chains.best <- eta.chains

        eta.sample1.1 <- eta.samp1.1
        eta.sample1.2 <- eta.samp1.2
        eta.sample1.3 <- eta.samp1.3
        eta.sample1.4 <- eta.samp1.4
        eta.sample1.5 <- eta.samp1.5

        eta.sample2.1 <- eta.samp2.1
        eta.sample2.2 <- eta.samp2.2
        eta.sample2.3 <- eta.samp2.3
        eta.sample2.4 <- eta.samp2.4
        eta.sample2.5 <- eta.samp2.5

        eta.vec.best      <- eta.vec
        eta.stat.best     <- eta.stat
        Xmat.best         <- Xmat

        Y_params.best  <- Y_params[c(1,2,(iter+2)), -c(4,7,9) , ]
        Z_params1.best          <- Z_params1[c(1,2,(iter+2)), c(1:3,8,9,11,13)]
        Z_params2.best          <- Z_params2[iter, 4:21]
```

```
                Ymeans.best     <- Ymeans
                Zmeans1.best    <- Zmeans1
                Zmeans2.best    <- Zmeans2

                Yvars.best              <- Yvars
                Zvars1.best             <- Zvars1
                Zvars2.best             <- Zvars2
                }

 #------------------------
# MONITOR CONVERGENCE 2 :
# PERFORM BRIDGE SAMPLING TO APPROX OBSERVED LIKELIHOOD
#------------------------
# record estimates for (k+1)th EM iteration
muY.b           <- muY
lambda.b        <- lambda
theta.b  <- theta
muZ.b           <- muZ
beta.b   <- beta
sigma.sq.b      <- sigma.sq
eta.chains.b <- eta.chains
```

```
if (bridge == TRUE && iter > 1)
{       # RUN bridge-sampling loop only when studying convergence
        #  and start from 2nd EM iteration
for (m in 1:M)
 {
 for (i in 1:N)
        {
        Y <- matrix(as.double(Ydata[i, ]))
        Z <- as.double(Zdata[i])
        eta.a <- eta.chains.a[i, m]
        eta.b <- eta.chains.b[i, m]
        Xm.a    <- matrix(Xmat[i, ])
        Xm.a[3]         <- eta.a
        Xm.b    <- matrix(Xmat[i, ])
        Xm.b[3]         <- eta.b

        Lik_aa[i, m]    <-      (1/sqrt(det(theta.a))) * (1/sqrt(sigma.sq.a)) *
                exp( -0.5 * (t(Y-muY.a-lambda.a%*%eta.a) %*% solve(theta.a) %*% (Y-
muY.a-lambda.a%*%eta.a) +
                1/sigma.sq.a * ((Z - t(Xm.a) %*% beta.a)^2 + penalty) +  eta.a^2 ))
```

```
        Lik_ab[i, m]    <-      (1/sqrt(det(theta.b))) * (1/sqrt(sigma.sq.b)) *

                exp( -0.5 * (t(Y-muY.b-lambda.b%*%eta.a) %*% solve(theta.b) %*% (Y-
muY.b-lambda.b%*%eta.a) +

                1/sigma.sq.b * ((Z - t(Xm.a) %*% beta.b)^2 + penalty) +  eta.a^2 ))


        Lik_ba[i, m]    <-      (1/sqrt(det(theta.a))) * (1/sqrt(sigma.sq.a)) *

                exp( -0.5 * (t(Y-muY.a-lambda.a%*%eta.b) %*% solve(theta.a) %*% (Y-
muY.a-lambda.a%*%eta.b) +

                1/sigma.sq.a * ((Z - t(Xm.b) %*% beta.a)^2 + penalty) +  eta.b^2 ))


        Lik_bb[i, m]    <-      (1/sqrt(det(theta.b))) * (1/sqrt(sigma.sq.b)) *

                exp( -0.5 * (t(Y-muY.b-lambda.b%*%eta.b) %*% solve(theta.b) %*% (Y-
muY.b-lambda.b%*%eta.b) +

                1/sigma.sq.b * ((Z - t(Xm.b) %*% beta.b)^2 + penalty) +  eta.b^2 ))

        }

 }

num1 <- apply(Lik_ab, 2, sum)

den1 <- apply(Lik_aa, 2, sum)

num2 <- apply(Lik_ba, 2, sum)

den2 <- apply(Lik_bb, 2, sum)

A <- sqrt(num1/den1)

B <- sqrt(num2/den2)

log_LR[iter] <- log(sum(A)) - log(sum(B))

}
```

```
# record estimates for (k)th EM iteration
muY.a           <- muY.b
lambda.a        <- lambda.b
theta.a  <- theta.b
muZ.a           <- muZ.b
beta.a   <- beta.b
sigma.sq.a      <- sigma.sq.b
eta.chains.a <- eta.chains.b
new.iter.best <- iter.best
if((iter - iter.best) == stop.iter)
        {
        iter.hi <- max(iter.hi, new.iter.best)
if (iter.hi > new.iter.best) target.replic <- replic   # identif replic with highest EM iteration
        break
        }
if (iter==maxiter)
        {
        break
        }
}       # end EM loop
#-----------------------------------------------------------------
#  END MCEM ITERATION: 2ND LOOP
#-----------------------------------------------------------------
```

```
line9 <- line9 + 1
#####################################################
# For the BEST EM iteration in jth replication:
#       Store all parameter estimates
#####################################################
# Record and update model fits & MCMC samples for best EM iteration
#       based on minimum total deviance
# means and std.dev of ESTIMATES

paramMeans1[1:p, replic]              <- c(Ymeans.best[1:p, 1])      # Y-intercepts
paramMeans1[(p+1):(2*p), replic]      <- c(Ymeans.best[1:p, 4])      # Y-lambdas
paramMeans1[(2*p+1):(3*p), replic]    <- c(Ymeans.best[1:p, 7])      # Y-thetas
paramMeans1[(3*p+1), replic]          <- Zmeans1.best[1]             # Z-threshold/intercept
paramMeans1[(3*p+2), replic]          <- Zmeans1.best[7]             # Z-grp.intercept
paramMeans1[(3*p+3), replic]          <- Zmeans1.best[10]            # Z-sigma.sq

paramVars1[1:p, replic]               <- sqrt(c(Yvars.best[1:p, 1]))  # Y-intercepts
paramVars1[(p+1):(2*p), replic]       <- sqrt(c(Yvars.best[1:p, 4]))  # Y-lambdas
paramVars1[(2*p+1):(3*p), replic]     <- sqrt(c(Yvars[1:p, 7]))       # Y-thetas
paramVars1[(3*p+1), replic]           <- sqrt(Zvars1.best[1])         # Z-threshold/intercept
paramVars1[(3*p+2), replic]           <- sqrt(Zvars1.best[7])         # Z-grp.intercept
paramVars1[(3*p+3), replic]           <- sqrt(Zvars1.best[10])        # Z-sigma.sq
```

```
paramMeans2[ , replic]          <- Zmeans2.best   # q spline coeff + 1 smoothing param (ubre)
paramVars2[ , replic]           <- Zvars2.best

# means and std.dev of STD ERRORS OF ESTIMATES
paramMeans1.se[1:p, replic]                     <- c(Ymeans.best[1:p, 2])       # Y-intercepts
paramMeans1.se[(p+1):(2*p), replic]   <- c(Ymeans.best[1:p, 5])       # Y-lambdas
paramMeans1.se[(2*p+1):(3*p), replic] <- NA                                     # Y-thetas
paramMeans1.se[(3*p+1), replic]                 <- Zmeans1.best[2]      # Z-threshold/intercept
paramMeans1.se[(3*p+2), replic]                 <- Zmeans1.best[8]      # Z-grp.intercept
paramMeans1.se[(3*p+3), replic]                 <- NA                           # Z-sigma.sq
paramVars1.se[1:p, replic]                      <- sqrt(c(Yvars.best[1:p, 2]))  # Y-intercepts
paramVars1.se[(p+1):(2*p), replic]              <- sqrt(c(Yvars.best[1:p, 5]))  # Y-lambdas
paramVars1.se[(2*p+1):(3*p), replic]  <- NA                                     # Y-thetas
paramVars1.se[(3*p+1), replic]          <- sqrt(Zvars1.best[2])         # Z-threshold/intercept
paramVars1.se[(3*p+2), replic]          <- sqrt(Zvars1.best[8])         # Z-grp.intercept
paramVars1.se[(3*p+3), replic]          <- NA                           # Z-sigma.sq

paramMeans2.se[ , replic]       <- Zmeans2.best   # q spline coeff + 1 smoothing param (ubre)
paramVars2.se[ , replic]        <- Zvars2.best

etaVectors[ ,replic]      <- eta.vec.best
```

```
# store sub-model deviances at EM covergence for each replication
all.deviances[replic, ]    <- round(c(Ymeans.best[1:p, 9], Zmeans1.best[12]), 1)


line10 <- line10 + 1
#if (replic == 3) stop("3rd replication completed")
}


#-------------------------------------------------
#  END REPLICATIONS: 1ST LOOP
#-------------------------------------------------


# CLOCK THE END OF EM ITERATIONS
End.time <- Sys.time()
Lapsed.time <- difftime(End.time, Start.time)
Lapsed.time


##############################################################################
#==========          END OF ALV MODEL RUN                    ============
##############################################################################
line1
line2
line3
line4
```

```
line5

line6

line7

line8

line9

line10


#=========================================

#          COMPILE REPLICATION RESULTS

#=========================================


##### COMPILE TRUE VALUES

p <- ncol(YZdata)-2       # let p = dim YZdata less (Z, GRP) -> no of Y variables

Parameter                 <- c(rep(c("Y-intercept","lambda","theta"), each=p),

                                  "Z-intercept", "group", "sigma^2")

Index               <- c(rep(1:p, 3), rep(1,3))

Pop_param <- c(muY.t, lambda.t, diag(theta.t), muZ.t, grp.interc.t, sigma.sq.t)

#Pop_se <- c(muY.se.t, lambda.se.t, theta.se.t, muZ.se.t, grp.interc.se.t, sigma.sq.se.t)

Pop <- data.frame(Parameter, Index, Pop_param)

Pop
```

```
#### PARAMETRIC COEFFICIENTS

L <- replic

Mean_Dev      <- matrix(apply(all.deviances, 2, mean), ncol=1) #  calculate col means of sub-model deviances

Mean_Est      <- matrix(apply(paramMeans1[,1:L], 1, mean), ncol=1) #  calculate row means of param estim

SD_Est <- matrix(apply(paramMeans1[,1:L], 1, sd), ncol=1) #  calculate row std dev

v.est <- matrix(apply(paramMeans1[,1:L], 1, var), ncol=1) #  calculate row variance


Mean_SE       <- matrix(apply(paramMeans1.se[,1:L], 1, mean), ncol=1) #  calculate row means of std.err of estim

SD_of_SE      <- matrix(apply(paramMeans1.se[,1:L], 1, sd), ncol=1) #  calculate row std dev

SE_by_SD <- Mean_SE/SD_Est

true <-  Pop[, 3]

Bias <- Mean_Est - true

RMS <-  v.est + Bias^2


pc <- data.frame(Pop,  round(data.frame(Mean_Est, SD_Est, Mean_SE, SE_by_SD, Bias, RMS), 3))

pc$Deviance <- round(c(Mean_Dev[1:5], rep(NA, 10), Mean_Dev[6], NA, NA), 1)

paramCoef  <- tt[-17, ]  # Remove GRP coeff

paramCoef
```

APPENDIX C: (CONTINUED)

```
# Save estimated eta vector for each replication

write.csv(etaVectors, file = "C:/.../*.csv", row.names = FALSE)


# Save results of replication studies

write.csv(paramCoef, file = "C:/.../*.csv", row.names = FALSE)


#################################################################

#          WHEN ALV MODEL IS FITTED TO A SINGLE DATASET,

#          COMPILE RESULTS FOR GAM COMPONENT AS FOLLOWS:

#################################################################

#===================================

# Plot the fitted curve - GAM

#===================================

# USE THE SELECTED BEST ETA ESTIMATE (AT EM CONVERGENCE)

data.comp                    <- YZdata

data.comp$eta   <- eta.vec.best

data.comp$eta_by_group <- eta.vec.best * YZdata$GRP

 fitZ.b   <- gam(Z ~ s(eta) + as.factor(GRP) + s(eta_by_group), family=Z.family,
data=data.comp)

summary(fitZ.b)

# save

write.csv(data.comp, file = "C:/…/ *.csv", row.names = FALSE)
```

APPENDIX C: (CONTINUED)

```
# ANALYTIC PLOT
lwd<-2; lwd2<-1;
Tx.col<-2; Ctr.col<-4;
fit <- fitZ.b$fitted.values
 group <- data.comp$GRP
risk <- data.comp$eta
ord <- order(risk)
yrange <- range(data.comp$Z)
xrange <- range(risk)

plot(risk, fit, type = "n",
#   main=  paste("Variation in Intervention Impact by Baseline Risk"),
           sub = "Vertical lines at risk percentiles",
  ylim=yrange,
  xlim=xrange,
  xlab = "Baseline Risk (eta)",
  ylab = paste("Distal Outcome (Z)"))                              # for continous Z
#   ylab = paste("Probability of Distal Outcome (Z)"))     # for binary Z

 xord <- risk[ord]
 fitord <- fit[ord]
 Grord1 <- group[ord]
```

```
lines(xord[Grord1 == 1 ],  fitord[Grord1 == 1 ], lty=1, lwd=lwd, col=Tx.col)

lines(xord[Grord1 == 0 ], fitord[Grord1 == 0 ], lty=2, lwd=lwd, col=Ctr.col)


tx.legend <- paste("Tx (n = ", sum(Grord1), ")")

ctrl.legend <- paste("Ctrl (n = ", sum( (Grord1 == 0) ),")")

legend(xrange[1],yrange[2], legend = c(tx.legend, ctrl.legend), lty=c(1,2), lwd=c(lwd, lwd),
col=c(Tx.col, Ctr.col))


Q <- matrix(quantile(risk, c(.25, .50, .75, .90, .95)))


segments(Q[1], yrange[1], Q[1], yrange[2], lty = 2)

segments(Q[2], yrange[1], Q[2], yrange[2], lty = 2)

segments(Q[3], yrange[1], Q[3], yrange[2], lty = 2)

segments(Q[4], yrange[1], Q[4], yrange[2], lty = 2)

segments(Q[5], yrange[1], Q[5], yrange[2], lty = 2)


text( Q[1], yrange[1], " 25th ", , adj = c(0,0), par(srt=90))

text( Q[2], yrange[1], " 50th ", , adj = c(0,0), par(srt=90))

text( Q[3], yrange[1], " 75th ", , adj = c(0,0), par(srt=90))

text( Q[4], yrange[1], " 90th ", , adj = c(0,0), par(srt=90))

text( Q[5], yrange[1], " 95th ", , adj = c(0,0), par(srt=90))
```

```
#-------------------------------------

# DIAGNOSTIC PLOTS

#-------------------------------------

gam.check(fitZ.b) # residual plots

plot(fitZ.b,pages=1,residuals=TRUE)

plot(fitZ.b,pages=1,seWithMean=TRUE, shade=TRUE)


#-----------------------------------------------------------------------

# MORE ALV MODEL FIT RESULTS FOR REVIEW

#-----------------------------------------------------------------------

Y_params.best

Z_params1.best

Z_params2.best


Y_params[1:(iter+2),-c(4,7,9) ,]

Z_params1[1:(iter+2),c(1:3,8,9,11,13)]


paramMeans1[,1:replic]
```

```
##############################################
#          ALV MODEL CONVERGENCE RESULTS
##############################################

# OVERAL FOR ALV MODEL
# record log of observed-data likelihood ratio between 2 consecutive steps
convergence$logLR.obs <- as.vector(log_LR)

# Create a new variable for total deviance minus its MINIMUM (for graphing purposes)
convergence$SumDeviance2 <- rep(NA, nrow(convergence))
mDev <- min(convergence$SumDeviance[1:iter])
convergence$SumDeviance2[1:iter] <- round((convergence$SumDeviance[1:iter] - mDev), 4)

# RECORD convergence data
convergence[1:iter, c(1:5,7)]
round(convergence[1:(iter-1), c(1:5)], 4)

# save
write.csv(convergence, file = "C:/.../*.csv", row.names = FALSE)
```

```
#-----------------------------------------------------------
# PLOTS TO MONITOR CONVERGENCE
#-----------------------------------------------------------

###      DEVIANCE PLOTS
par(mfrow = c(2, 1))

plot(c(1:iter), convergence$logLR.obs[1:iter], type="l",
        ylab="Log of likelihood ratio", xlab="Iteration")
        title(main="Log of Observed-Data Likelihood Ratio
                Versus EM Iteration from the 2nd Iteration", cex.main=1.1)
abline( v = iter.best, col = "blue", lty=3)

plot(c(1:iter), convergence$SumDeviance2[1:iter], type="l",
        ylab="Total deviance", xlab="Iteration")
        title(main="Scaled Total Deviance Versus EM Iteration", cex.main=1.1)
abline( v = iter.best, col = "blue", lty=3)
```

```
###       CONVERGENCE ERRORS
par(mfrow = c(1, 1))
plot(convergence[,1], convergence[,3], ylim=c(0,3), xlim=c(0,iter),type="n",
          ylab="Convergence Errors", xlab="Iteration")
          title(main="Log of Observed-Data Likelihood Ratio
                    Versus EM Iteration from the 2nd Iteration", cex.main=1.1)
#abline( v = iter.best, col = "black", lty=3)
lines(convergence[,1], convergence[,3], lwd=1.9, lty=2, col=1)
lines(convergence[,1], convergence[,4], lwd=1.9, lty=1, col=1)

err1.legend <- paste("{mu's, lambda's, theta's, sigma^2's}")
err2.legend <- paste("beta's")

legend("topright", legend = c(err1.legend, err2.legend), lty=c(2, 1),
                  horiz = FALSE, lwd=c(1, 1), col=c(1,1))

#-----------------------------------------
# MONITOR PARAMETER ESTIM
#-----------------------------------------
yp <- Y_params[1:(iter+2),-c(4,7,9) ,]
zp1 <- Z_params1[1:(iter+2),c(1:3,8,9,11,13)]
 est1 <- est2 <- est3 <- matrix(NA, ncol=p, nrow=(iter+1))
```

```
for (k in 1:p)
{
est1[ , k]          <- as.double(yp[c(2:(iter+2)),2 ,k])        # y-interc
est2[ , k]          <- as.double(yp[c(2:(iter+2)),4 ,k])        # lambda
est3[ , k]          <- as.double(yp[c(2:(iter+2)),6 ,k])        # theta
}

est4 <- matrix(NA, ncol=3, nrow=(iter+1))
est4[,1] <- as.double(zp1[c(2:(iter+2)),2]) # z-interc
est4[,2] <- as.double(zp1[c(2:(iter+2)),4]) #  grp
est4[,3] <- as.double(zp1[c(2:(iter+2)),6]) # sigma.sq
estA <- data.frame(cbind(est1, est2, est3, est4))
rm(yp)
rm(zp1)
names(estA)  <- c("Y1", "Y2", "Y3", "Y4", "Y5", "lam1", "lam2", "lam3", "lam4", "lam5",
                  "the1", "the2", "the3", "the4", "the5", "Z", "grp","sig2")

par(mfrow = c(2, 2))
iteration <- 0:iter
```

```
plot(iteration, estA[,1], ylim=range(estA[,1:5]), ylab="estimate", type="n")
        title(main="Y-intercepts", cex.main=1.1)
abline( v = iter.best, lty=3)
for (j in 1:5)
{
lines(iteration, estA[,j], col=(j+1), lty=(j+1), lwd=2)
}

plot(iteration, estA[,1], ylim=range(estA[,6:10]), ylab="estimate", type="n")
        title(main="Lambdas", cex.main=1.1)
abline( v = iter.best, col = "blue", lty=3)
for (j in 6:10 )
{
lines(iteration, estA[,j], col=(j-4), lty=(j-4), lwd=2)
}
plot(iteration, estA[,1], ylim=range(estA[,11:15]), ylab="estimate", type="n")
        title(main="Thetas", cex.main=1.1)

abline( v = iter.best, col = "blue", lty=3)
for (j in 11:15 )
{
lines(iteration, estA[,j], col=(j-9), lty=(j-9), lwd=2)
}
```

```
 plot(iteration, estA[,1], ylim=range(estA[,c(16,18)]), ylab="estimate", type="n")
          title(main="Sigma^2 -.-   Z-interc __", cex.main=1.1)
abline( v = iter.best, col = "blue", lty=3)
lines(iteration, estA[,16], lwd=2, lty=2)
#lines(iteration, estA[,17], lwd=2, lty=3)
lines(iteration, estA[,18], lwd=2, lty=4)

#-------------------------------------------------------------------------------------------
# PLOTS to examine MCMC simulations for 5 randomly selected subjects
#-------------------------------------------------------------------------------------------
plot(eta.sample1.1)
mtext("1st Randomly Selected Subject (GRP=1)", cex = 1.2, side = 3, line = 3)

plot(eta.sample1.2)
mtext("2nd Randomly Selected Subject (GRP=1)", cex = 1.2, side = 3, line = 3)

plot(eta.sample1.3)
mtext("3rd Randomly Selected Subject (GRP=1)", cex = 1.2, side = 3, line = 3)

plot(eta.sample1.4)
mtext("4th Randomly Selected Subject (GRP=1)", cex = 1.2, side = 3, line = 3)
```

```
plot(eta.sample1.5)
mtext("5th Randomly Selected Subject (GRP=1)", cex = 1.2, side = 3, line = 3)

plot(eta.sample2.1)
mtext("1st Randomly Selected Subject (GRP=2)", cex = 1.2, side = 3, line = 3)

plot(eta.sample2.2)
mtext("2nd Randomly Selected Subject (GRP=2)", cex = 1.2, side = 3, line = 3)

plot(eta.sample2.3)
mtext("3rd Randomly Selected Subject (GRP=2)", cex = 1.2, side = 3, line = 3)

plot(eta.sample2.4)
mtext("4th Randomly Selected Subject (GRP=2)", cex = 1.2, side = 3, line = 3)

plot(eta.sample2.5)
mtext("5th Randomly Selected Subject (GRP=2)", cex = 1.2, side = 3, line = 3)

############################
#         SAVE WORKSPACE
############################
save.image(file = "C:/…/*.RData")
```

ABOUT THE AUTHOR

Peter Toyinbo received a doctoral Degree in Medicine from University of Ife, Nigeria in 1981 and an M.S.P.H. (Biostatistics) from University of South Florida (U.S.F.) in 2004. He was accepted into the Ph.D. program at the U.S.F. in 2004. Concurrently he completed a two-year postdoctoral fellowship in Preventive Research Methodology in Mental Health from the Johns Hopkins Bloomberg School of Public Health in 2006. He received the national 'Certified in Public Health' (CPH) credential in 2009.

He is an active member of the Prevention Science and Methodology Group (PSMG) and has collaborated on few research projects through the PSMG research efforts. He currently holds the position of Statistical Data Analyst with the Department of Aging and Mental Health Disparities in the Florida Mental Health Institute of the USF where he collaborates and provides consultations on research study designs and statistical analyses to fellow research faculty.